# RANGE DATA HANDLING INTEGRATION STUDIES

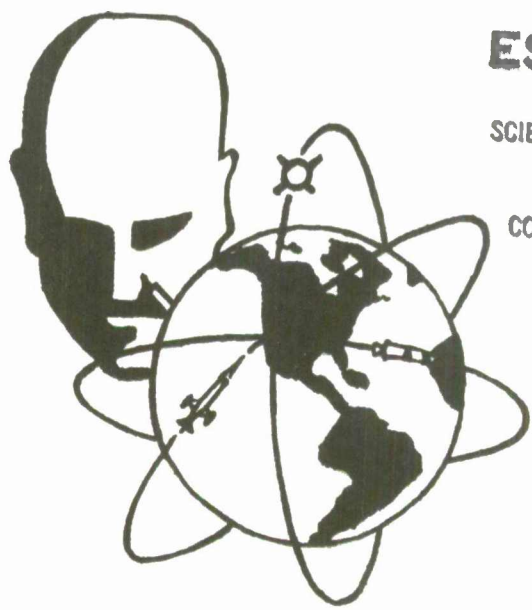
## IN-LINE TRANSLATORS INCORPORATING BINARY / DECIMAL CONVERTERS

*Walter D. Urban*

NOVEMBER 1964



DIRECTORATE OF AEROSPACE INSTRUMENTATION
ELECTRONICS SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts





Project 5932 Task Number .01
MIPR 4230486

DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington 25, D. C.

NBS PROJECT
4230486

NBS REPORT
8578

IMPORTANT NOTICE

NATIONAL BUREAU OF STANDARDS REPORTS are usually preliminary or progress accounting documents intended for use within the Government. Before material in the reports is formally published it is subjected to additional evaluation and review. For this reason, the publication reprinting, reproduction, or open-literature listing of this Report, either in whole or in part, is not authorized unless permission is obtained in writing from the Office of the Director, National Bureau of Standards, Washington 25, D.C. Such permission is not needed, however, by the Government agency for which the Report has been specifically prepared if that agency wishes to reproduce additional copies for its own use.


Qualified requesters may obtain copies from Defense Documentation Center (DDC). Orders will be expedited if placed through the librarian or other person designated to request documents from DDC.

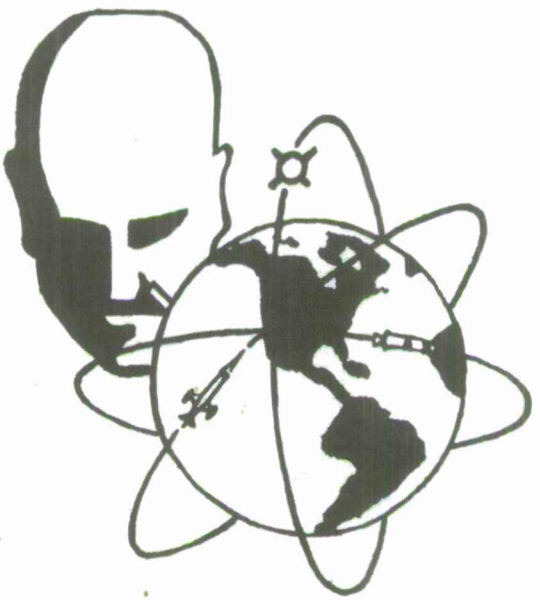
# RANGE DATA HANDLING INTEGRATION STUDIES

## IN-LINE TRANSLATORS INCORPORATING BINARY / DECIMAL CONVERTERS

*Walter D. Urban*

NOVEMBER 1964

DIRECTORATE OF AEROSPACE INSTRUMENTATION
ELECTRONICS SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
UNITED STATES AIR FORCE
L. G. Hanscom Field, Bedford, Massachusetts



Project 5932 Task Number .01

MIPR 4230486

DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington 25, D. C.

# FOREWORD

This document has been prepared by the Data Processing Systems Division of the National Bureau of Standards under Task 5932.01 for the Air Force Electronic Systems Division in support of Hq AFSC PD 650A. It is the final report of the study broadly entitled "In-Line Translators." In actuality the scope is much more limited since devices properly called "in-line" translators (matrix converters) are in use throughout the data network in great numbers. However, none of these are being used for radix conversion. This report is concerned with in-line translators for radix conversion which do not yet exist, and which can find profitable application in the specific instance of data message conversion.

The purpose of the study has been to determine the feasibility of applying this technique to data networks, to determine how best to design such devices and to establish designs. It has also been our purpose to uncover heretofore unrealized problems of design and application of these devices.

Textual preparation, assistance, editing, and printing were done by Edward T. Johnson and Associates of Washington, D. C.

Arthur A. Ernst
Project Manager

## ABSTRACT

The application of In-Line Translators is covered in terms of the specific adaptation of a wired program device which can efficiently handle the translation between different numerical radices or number base codes. Design criteria, feasibility, and a functional description are presented.

Since the Radix Converter is the central and primary modular element of the translator concept, the logic designs for four principal types of decimal/binary converters are described in detail.

TABLE OF CONTENTS

## LIST OF ILLUSTRATIONS

## Section 1. INTRODUCTION

### 1.1 GENERAL.

1.1.1 A particular problem in the integration and augmentation of existing information handling networks is the inconsistency in the format of data passed between various points in the data network. At times of peak data loads when data equipment is working at near capacity, these incompatibilities have the potential of causing major inefficiencies to occur. Format disparities can be classified as those resulting from the message composition, the code symbolism utilized, and the physical characteristics associated with transmission and receipt. (Figure 1)

| TYPE OF FORMATTING | STANDARDIZATION | TRANSLATION | |
|---|---|---|---|
| | | IN USE | PREFERRED |
| COMPOSITION | MESSAGE STRUCTURE AND CONTENT | WIRED PROGRAM | STORED PROGRAM |
| CODE | CHARACTERS | MATRIX CONVERTERS TABLE LOOK UP | EITHER |
| | NUMBER BASE | STORED PROGRAM | WIRED PROGRAM |
| PHYSICAL | PULSE LENGTH MODULATION | TERMINAL LOGIC MODEMS | SAME |

FIGURE 1. SIX KINDS OF FORMATTING PROBLEMS

1.1.2 The problem of message data incompatibility can be solved all or in part by an all encompassing standardization, by wired programs, by stored programs, or by a combination of these. However, when considered one at a time, none of these solutions lend themselves to the problem as the one best solution.

1.1.3 A wholesale standardization of formats is a difficult process to implement when consideration is given to factors such as the immediate outmoding of equipment which would result, the attendant huge problem of revising procedures throughout the data network complex, and the time and cost that would be required to change over to new procedures and equipment.

1.1.4 Physical formatting translation is necessarily handled by Terminal Logic with conventional modulation and demodulation devices. Code character translation can be handled readily either through the use of Matrix Converters or Table Look-Up Techniques.

1.1.5 Composition inconsistencies in message structure and content can and are being rectified through the use of wired program devices. However, in view of the large number of differently configured special devices required to obtain network compatibility, it seems reasonable that stored programs would be better suited for this purpose. Conversely, the translation between different numerical radices or number base codes are being handled by stored programs when it could be done more effectively by wired program converters where the work load justifies the initial hardware cost.

1.1.6 It is the purpose of this study to demonstrate that this latter problem can be efficiently handled by means of wired logic converters and that such converters can be useful in reaching the goal of adequate real-time operation and can offer a significant economic advantage for heavy data flow over data conversion by means of stored programs.

## 1.2 OBJECTIVE.

It is the objective of this study to:

(1) Establish whether satisfactory translators can be designed to perform specified routine in-line translation for high volumes of data.

(2) Establish whether construction of such translators is feasible in terms of cost and as a useful augmentation to currently installed computers.

(3) Establish the basic characteristics required of in-line translators.

(4) Establish design criteria.

(5) Develop logic designs of BCD to binary and binary to BCD converters.

(6) Develop logic designs of time in BCD to binary and time in binary to BCD converters.

## 1.3 APPROACH.

The translator when interposed between a particular data source

and receptor must be capable of accepting the data signals, converting the data and generating signals suitable for use by the data receptor. The approach taken in this study has been first to analyze the translator as an entity in the data system. Following this, the data conversion problem has been removed from its system interface in order to assure a radix converter design which is readily adaptable for use throughout the data network. This can be achieved if the data for conversion is preconditioned and standardized by an input unit of the translator prior to entry into the radix converter. After conversion, the data is organized for transmission to the data receptor by an output unit. Thus, the translator has been separated into an Input Buffer, Composition Controller, Radix Converter, and an Output Buffer. Logic designs of the converter have been developed as a part of the study; a design of the complete translator can be developed for any specific application for which a given set of values for input data and the characteristics of the data line are stipulated.

## Section 2. DESIGN CRITERIA

### 2.1 RELIABILITY.

2.1.1 The system reliability of a data stream is equal to the product of the reliabilities of all the units in the stream. Therefore, the reliability of each element placed in series is of particular importance. A malfunction in an in-line unit of a data stream can alter or even destroy data passing through it. For this type of function, reliability should only be limited by the feasibility of construction and economic practicality.

2.1.2 Current techniques in the utilization of solid state components lend themselves to achieving this end. The use of components which require regular maintenance, frequent performance checks, and adjustment should be avoided. Similarly, the circuits employed must follow good design techniques, be held to a minimum, and be relatively insensitive to power line fluctuations or to disruptive signals which may be generated by neighboring devices.

2.1.3 The device should be capable of sustained operations with a minimum amount of maintenance. Design of the unit should include careful consideration of the need for the ready detection of malfunctions and ease of repair.

### 2.2 CONVERSION SPEED.

The translator must not introduce a data lag any greater than now experienced and should, for the sake of system improvement, actually improve on current processing times. It is preferable to have unused capacity at a minor additional cost than to require a particular circuit design for every speed requirement; therefore, translators are to be designed for the highest practical speed of operation within the limits of reasonable costs and currently available solid state circuitry. The designed processing limit will be 100,000 six bit characters per second which is at about the ceiling of the current magnetic tape transport capability.

### 2.3 PRECISION

Translators are to be designed to accomodate a precision rate of one part per billion. Included will be a provision for operating at a lower precision of one part per million by a simple adjustment and/or deletion of components.

2.4 ADAPTIVENESS.

The design of the translator should permit a maximum amount of flexibility in application. This built-in adaptivity will obviate the necessity of designing a special device for every translator application.

2.5 INDEPENDENCE.

The translator is to be a self-contained independent entity capable of being inserted in the data line and functioning thereafter without further external assistance.

## Section 3. IN-LINE TRANSLATORS

### 3.1 FUNCTIONAL DESCRIPTION.

In actuality, the translator will be composed of approximately ten modular units. However, for purposes of explanation, consider that the translator is composed of four principal elements; an Input Buffer, a Composition Control Unit, a Radix Converter, and an Output Buffer. (Figure 2a and 2b) The following is a basic description of the processes occurring in these elements.

3.1.1 Input Buffer. Since the required translation could include changes in the physical, code, and composition format, parity checks, and data flagging, the radix conversion may only be a part of the complete translation process. For this reason it is desirable to make the converter "system independent." The first step in achieving this in the interception of the data from the data link by an input unit which alters the incoming message to make it acceptable for introduction into the Composition Control Unit. In this input unit there may be a demodulator which puts the proper physical format on the received signal for further passage in the translator. Thus, the input unit is capable of accepting the input signal, physically adjusting, and storing the input message.

3.1.2 Composition Control Unit. This is a wired program device which essentially integrates and controls the operations of the translator. It controls the transfer of measurement data into the radix converter and controls the composition of the output message. The incoming message procedural data, such as headers and flags, are separated from the measurement data by an implicit or explicit wired program and stored for later recombination after data conversion. If there is to be a data quality check it will be done on receipt in the Composition Control Unit. Also in this unit the measurement data is assembled and restructured as necessary for code and radix conversion. The measurement data is separated into characters or syllables for introduction into the radix converter. However, these characters may not be in the code for which the converter was designed and it may therefore be necessary to insert an additional step before the data conversion process. If the code is incompatible, a code standardization operation will be provided. This step may be required, for example, in converting data from BCD to binary. At the completion of this process all data entered into the radix converter will be in the form of 8-4-2-1 characters called BYTES. Because the number of bits of translated data will not agree with the number sent into the translator, the Composition Control portion must either make it possible to add or delete characters from the string if they are received and produced serially, or to enter blank characters

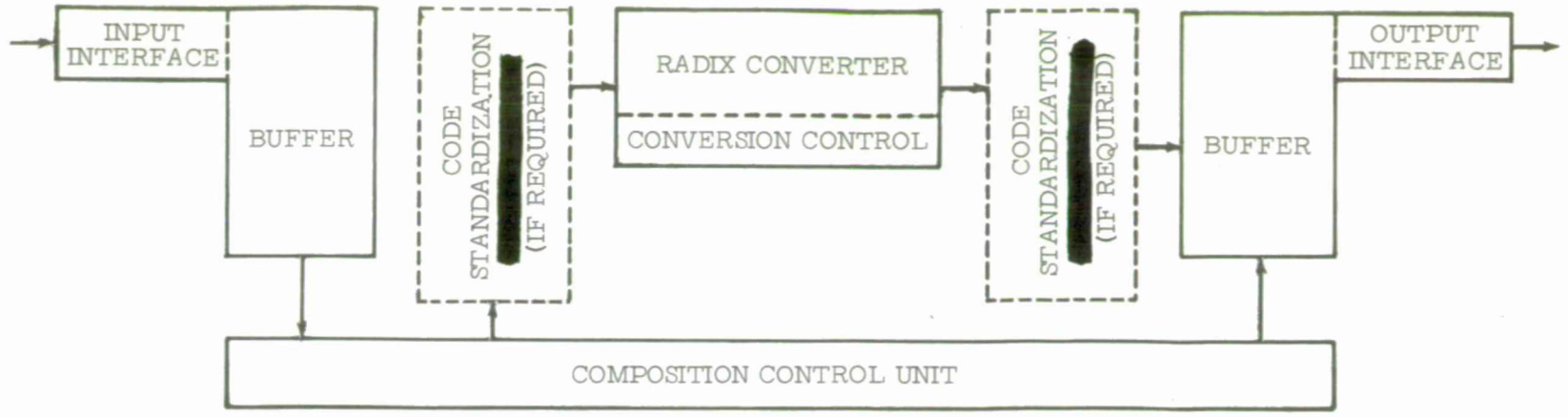



FIGURE 2a. FUNCTIONAL DIAGRAM OF A TYPICAL IN-LINE TRANSLATOR

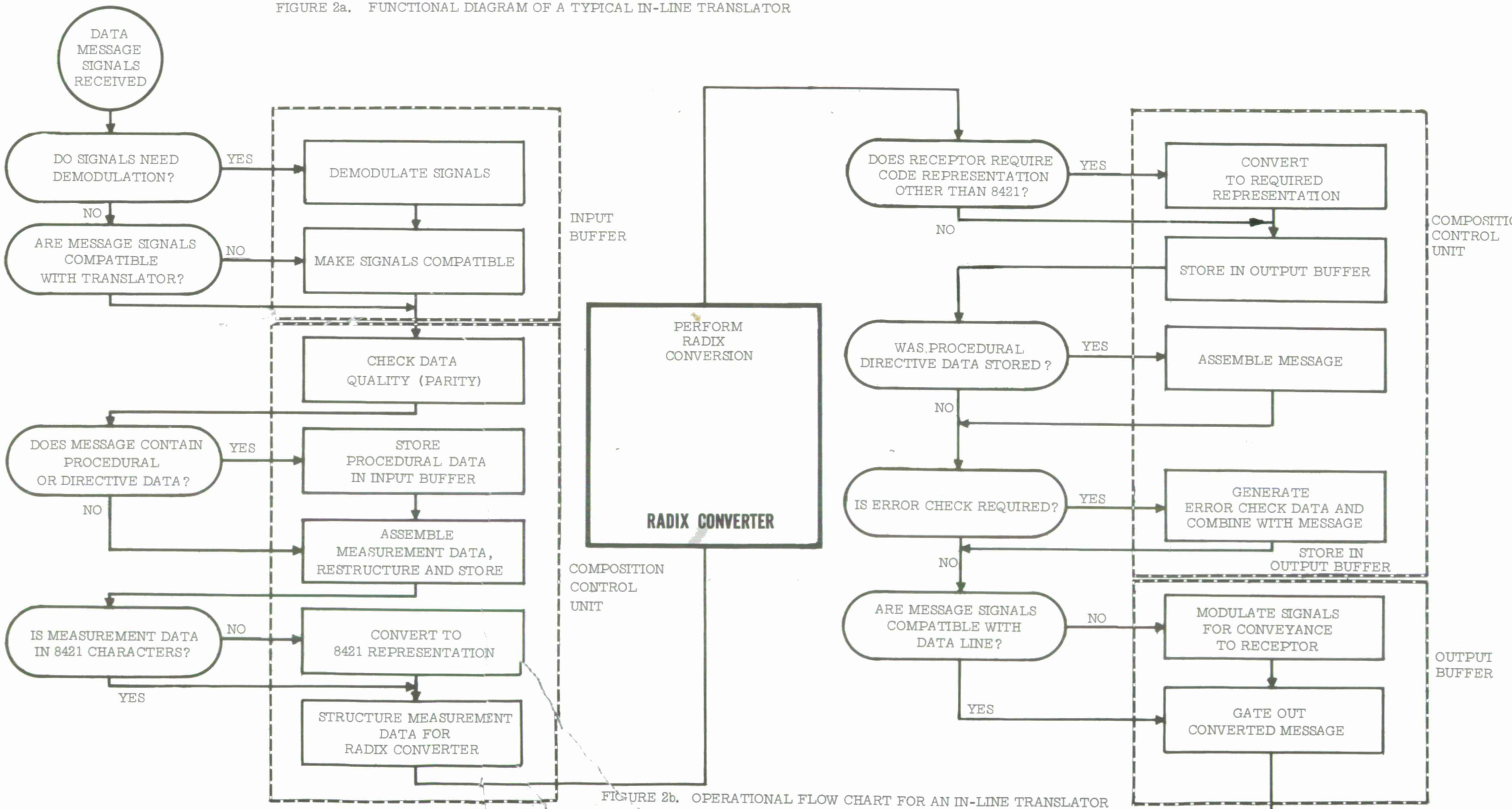



FIGURE 2b. OPERATIONAL FLOW CHART FOR AN IN-LINE TRANSLATOR

where a fixed parallel format is used. When the measurement data is received in a form unsuitable for convenient conversion, the incoming data characters are assembled and stored in the input buffer until the complete measurement data is available. It is then restructured into the proper form for conversion by the Composition Control Unit. This step will cause a time lag in addition to the delay caused by the conversion process which is equal to the time required to receive the entire measurement word. Since the tasks of these units may vary considerably from use to use, it may be necessary to provide a variety of such modules for translators.

3.1.3 Radix Converter. The radix conversion will be accomplished by a wired-program unit. A minimum of storage operations will be included in order to keep conversion speed high. Arithmetic operations are accomplished, as much as possible, by stream methods in which the output responds to the inputs within a settling-down-time which is small compared to an incoming character period. For example, in the BCD to binary converter, the time required to handle each incoming character is expected to be about 6 microseconds. The Radix Converter is covered in further detail in Section 4.

3.1.4 Output Buffer. The output interface presents a problem similar to that of the input interface. Here the output unit provides the necessary storage, places the converted data in the correct code format, and under the control of the Composition Control Unit, places the outgoing message in the proper composition format. Finally, the interface section of the output unit places the outgoing signal in the required physical format. The physical format change may require a modulator; the code format change may require a wired matrix to go from 8-4-2-1 to BCD or whatever code is used by the receptor. The Composition Control and Output Unit combine to keep the flow of characters or words proceeding at the rate at which they enter so as to obviate the need for storage of data or to minimize the delay through the translator. Since not all data in a message are converted, some may be bypassed onto the line while the rest goes through the converter. The translator control unit handles this type of traffic pattern, allowing only data to be converted to go to the converter.

## 3.2 DESIGN FEASIBILITY.

From an analysis of a wide range of variable input and output requirements and the structure of the designs associated with these requirements, the general problems associated with the design of in-line translators have been organized into a manageable form. As a consequence, it has been determined that the design and construction is technically feasible within the present state of the art, and that the elements required for construction are commercially available at reasonable cost.

## 3.3 EMPHASIS ON RADIX CONVERTER DESIGN.

The premise throughout this study has been to design a translating device which can be readily inserted at any required point in the data network with a minimum of re-configuration. As previously indicated, the wide range of possible data requirements and associated variables have been considered in the determination of design feasibility. The study has demonstrated that it is highly practical to meet the particular requirements of a given application by the addition, deletion, or restructuring of modular units of a basic translator. However, until the specific input/output values are stipulated for a point of application, any detailed design of the entire translator unit would be purely an academic exercise. Primary design effort has therefore been concentrated on the Radix Converter. This unit is the core of the translator and is designed to have unlimited application for a specific type of conversion regardless of the individual interface requirements of the complete translator.

## Section 4. RADIX CONVERTERS

### 4.1 GENERAL.

To relieve the central data processors of part of the data conversion burden, it is proposed to provide several types on in-line translators. In particular, four types are being considered:

(1) Translation of data from BCD to binary.
(2) Translation of data from binary to BCD.
(3) Translation of data from time in hours, minutes, seconds, and milliseconds to binary milliseconds.
(4) Translation of time in binary milliseconds, to hours, minutes, seconds, and milliseconds.

As previously explained, the heart of this translation process is in the Radix Converter. This section outlines in detail the design parameters and the logic design in the four types of converters.

### 4.2 DESIGN PARAMETERS AND GENERAL DESCRIPTION.

4.2.1 The converters are designed to handle pure measurement data up to 100,000 six bit characters per second. At lower input rates the converter has been designed to convert the incoming data BYTE at its maximum rate, store the results and await the next BYTE to be processed. Higher data rates will be accommodated by the use of parallel conversion for which converters have been designed.

4.2.2 A data precision of one part per billion or a number containing nine decimal digits or 30 bits has been designed into the converter. For cost and circuit comparison purposes, circuits for a lower precision rate of one part per million have also been designed. Some economy can be achieved where very short numbers are to be handled by omitting some of the circuitry required for the higher precision while retaining the basic universal module.

4.2.3 To further promote design simplicity, wherever possible the data measurements are considered to be in terms of the smallest unit employed, thus making the entire measurement a whole number. For example, hours, minutes, and seconds would each be expressed in milliseconds.

4.2.4 Two basic types of conversion methods have been developed. The first of these (serial conversion) converts the data measurement piecemeal serially by character or BYTE, using iterative operations. The second (parallel conversion) operates on the entire data measurement in a systematic way in a single operation. Being more economical of

circuitry, the serial conversion will generally be used even when the incoming data is received broadside. However, short processing time requirements may demand that parallel conversion be utilized. Figures 3 and 4 demonstrate that the circuitry required for parallel operations greatly exceeds that needed for serial operation for precision capabilities in excess of one part per 100,000.

4.3 CONVERSION OF DATA FROM BCD TO BINARY.

4.3.1 Let $D_1\ D_2\ D_3\ D_4$ represent a 4 decade decimal number. The rank of each digit is expressed by writing the number in its explicit form:

$$D_1\ X1000 + D_2\ X100 + D_3\ X10 + D_4$$

Rewriting this number in the equivalent nested form demonstrates the decimal to binary process:

$$D_1\ D_2\ D_3\ D_4 = 10\,(10\,(10D_1 + D_2) + D_3) + D_4$$

Again, consider a number whose digits are:

$$D_1\ D_2\ D_3\ D_4$$

$D_1$ is the most significant digit and $D_4$ is the least significant. Using binary operations, $D_1$ is multiplied by ten (1010) and $D_2$ is added to the product to obtain the intermediate sum $S_1$. $S_1$ is in turn multiplied by ten and $D_3$ is added to this product to obtain the next intermediate sum $S_2$. The process is updated to obtain the sum $S_3$ which is the required binary equivalent of $D_1\ D_2\ D_3\ D_4$. This process is summarized as follows:

$$10\ D_1 + D_2 = S_1$$
$$10\ S_1 + D_3 = S_2$$
$$10\ S_2 + D_4 = S_3$$

4.3.2 In the serial conversion process, each digit is processed in a separate cycle. Thus the determination of each intermediate sum is a separate conversion step in the iterative process. The parallel conversion process, on the other hand, involves a single stream operation on a chain of adders. In our example, digits $D_1$ and $D_2$ are combined to form the sum $S_1$ which is immediately thrust forward to be combined with $D_3$ to form the sum $S_2$. The process continues uninterrupted until the conversion is complete. Actual multiplication is avoided; instead, multiplication by ten is accomplished by adding twice the multiplicand to eight times the number. This process involves binary shift operations.

4.3.3 Figures 5 and 6 are relevant to the exposition which follows. For serial operations the converter consists of an Augend Register, an Addend Register, a Sum Buffer, stream binary adders, a Data Switch, and control circuitry. To simplify the latter, each cycle processes only one digit. With each new value to be converted a signal is included which clears the Augend and Addend Registers.

4.3.4 When the first (most significant) BCD 4 bit BYTE is loaded into

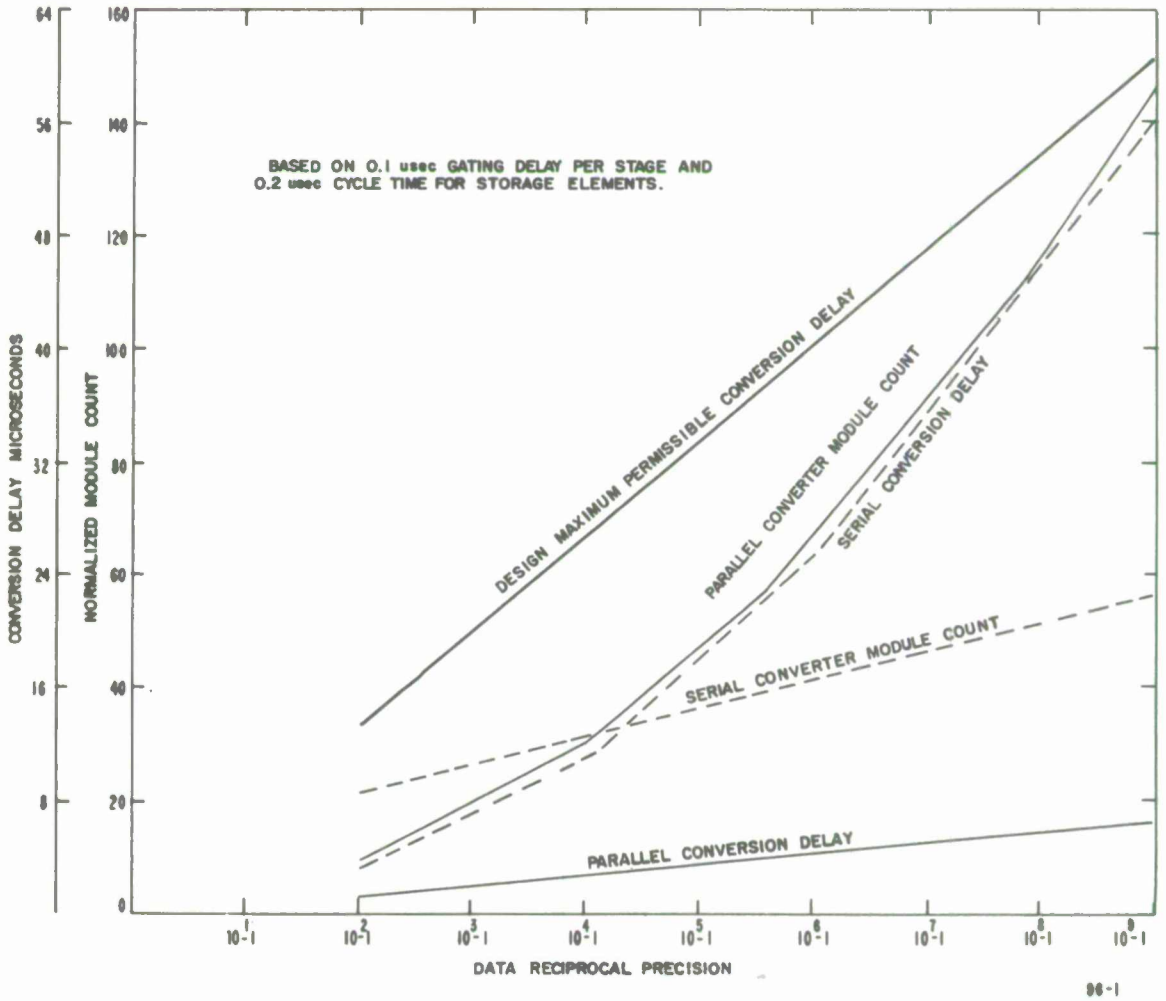



FIGURE 3. BCD TO BINARY CONVERTER COMPARISONS OF REQUIREMENTS AND CONVERSION DELAYS AS FUNCTION OF PRECISION CAPABILITY.

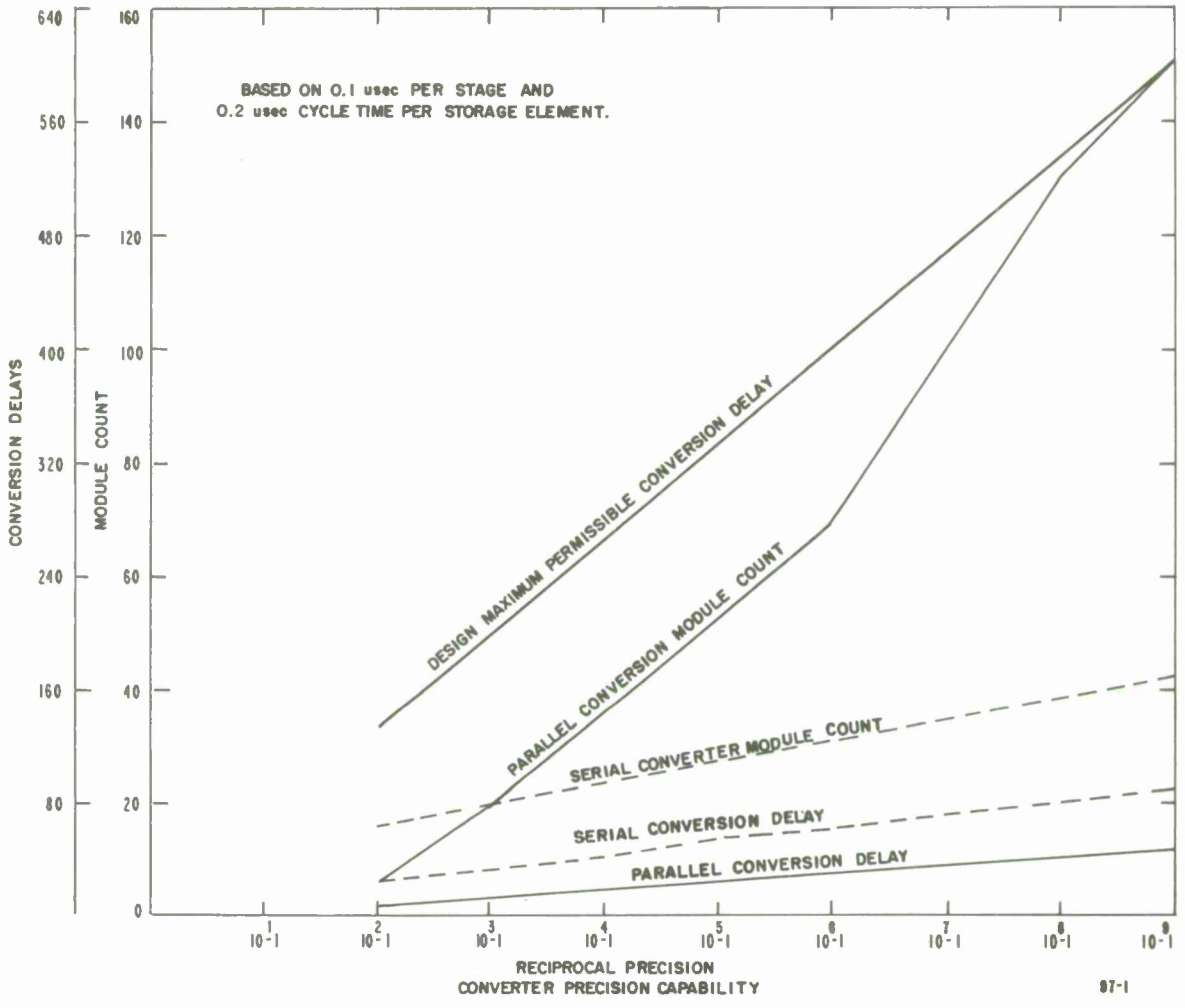
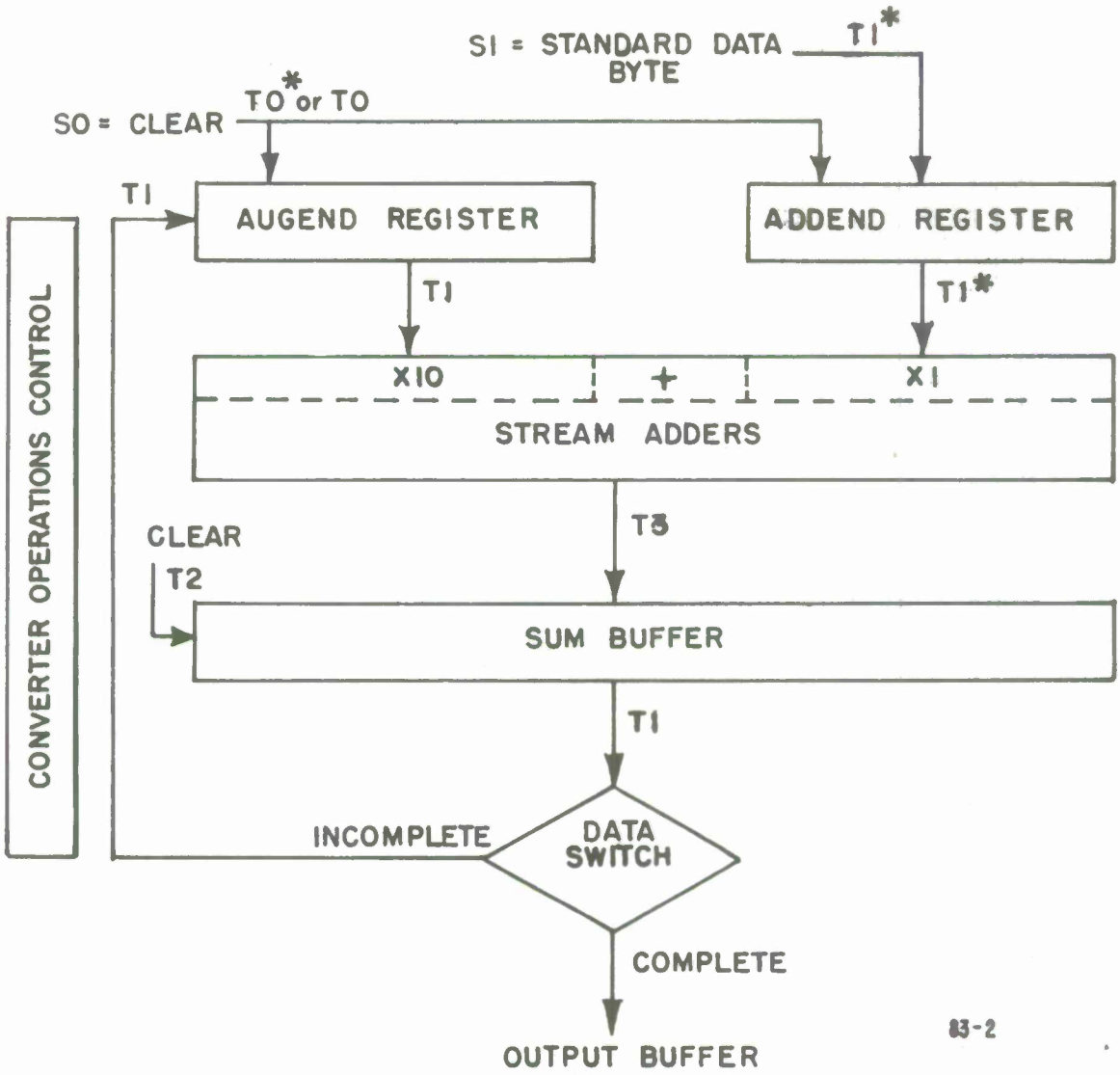



FIGURE 4. BINARY TO BCD CONVERTER COMPARISONS OF REQUIREMENTS AND CONVERSION DELAYS AS FUNCTION OF PRECISION CAPABILITY.




SERIAL DATA STREAM BCD- BINARY CONVERTER

FIGURE 5.

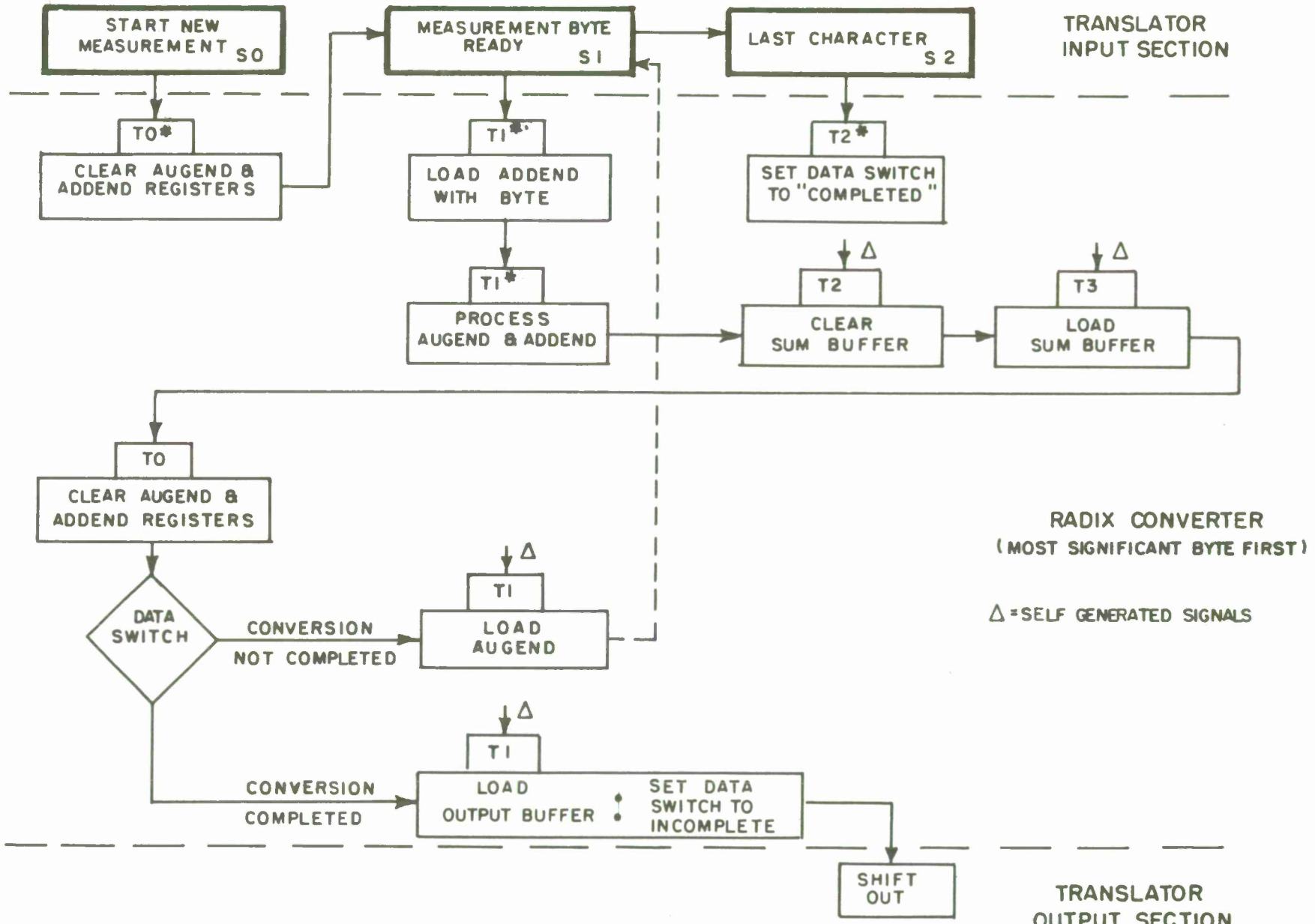
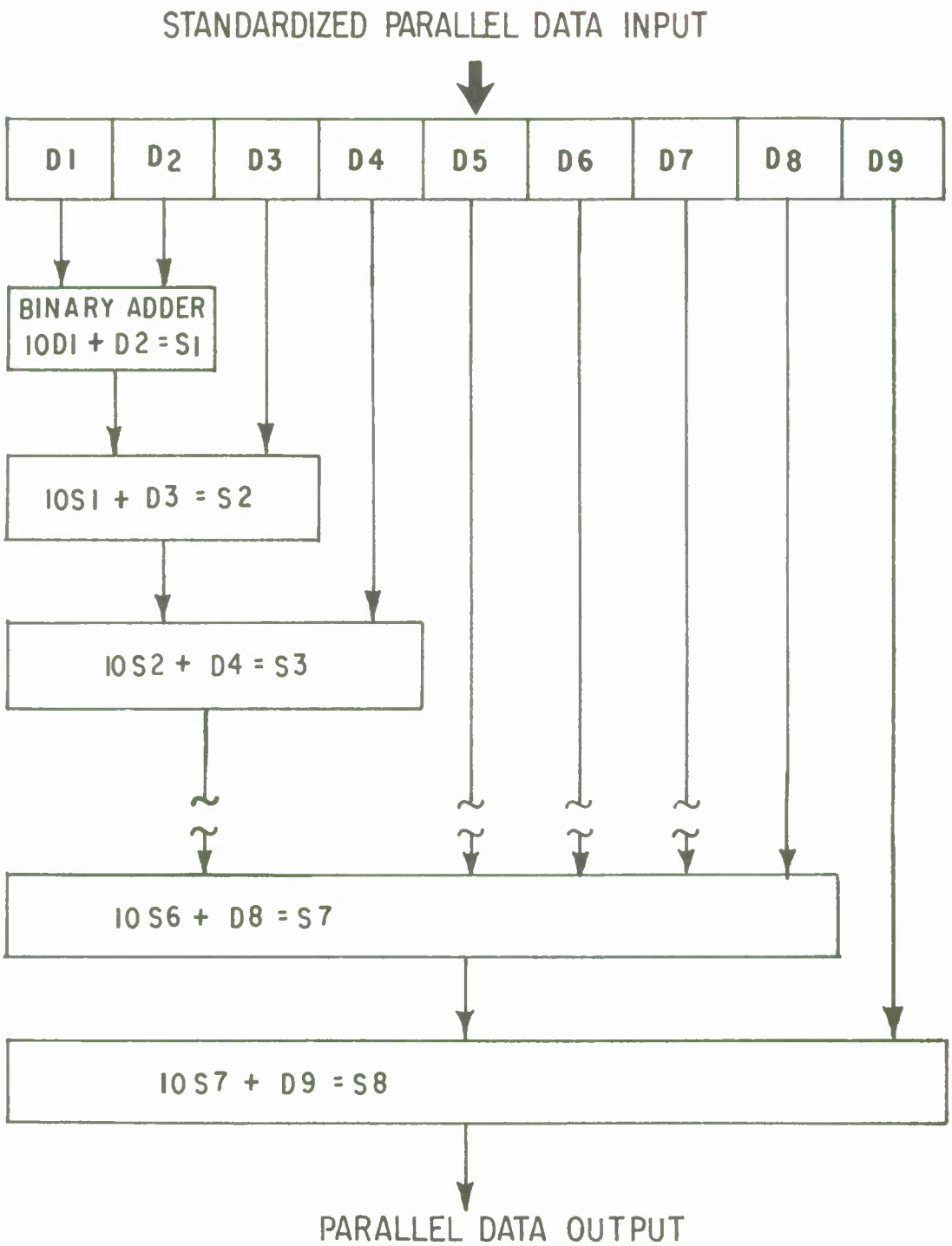



FIGURE 6. SEQUENCE OF EVENTS FOR CONVERTING DATA SERIALLY BY BYTES 98-1

the Addend Register the converter performs the first cycle of the conversion process at its designed rate. Using binary operations, each cycle consists of multiplying the contents of the Augend Register by ten and adding the contents of the Addend Register; this sum is temporarily stored in the Sum Buffer and then transferred to the Augend Register. The cycle is repeated with each remaining most significant digit until the conversion is complete.

4.3.5 The time sequences of events (Figure 6) in the conversion of BCD numbers to binary commences with the arrival of the signal $S_0$ signifying that a new measurement is ready for processing. $S_0$ clears the Augend and Addend Registers by allowing time $T_0$ to be internally generated. Next, when the first measurement BYTE is ready for processing, the signal $S_1$ permits the Addend Register to be loaded with the BYTE which is then processed with the value being held in the Augend Register. However, since this is the first BYTE of a new measurement the Augend Register is empty and the value is passed to the pre-cleared Sum Buffer at time $T_3$ where it is held pending the next internally generated $T_1$. The next generated $T_0$ clears the Addend and Augend Registers and at the following self-generated $T_1$ time the Augend Register is loaded with the contents of the Sum Buffer through the Data Switch. If the next input signal $S_1$ does not appear at this time, operations are suspended pending its receipt. In this way the converter adapts itself to any input rate. When $S_1$ appears the next most significant BYTE is introduced into the Addend Register where it is added to ten times the value of the contents of the Augend Register. While this process is still going on, the generated $T_2$ causes the Sum Buffer to be cleared and $T_3$ allows the new summation to be loaded in the Sum Buffer. Again, as the cycle re-commences at $T_0$, the Augend and Addend Registers are cleared of the old values and at $T_1$ the new value is placed in the Augend Register. The process continues as before until the last BYTE is ready for processing. At the signal $S_2$ indicating the final BYTE, the Data Switch is shifted to "Conversion Completed" and the converted value is loaded into the Output Buffer. The converter then halts to await the next Start of Message Signal.

4.3.6 Data presented in parallel can be converted by the device outlined in Figure 7, which uses the same mathematical principal as used in the serial BCD conversion. This device contains a 36 bit register plus eight groups of stream binary adders that process nine decade decimal numbers. Referring to the figure, the data word is stored in the Register. The number then has its BCD (BYTES) presented to the adder groups which follow. Thus the most significant digit $D_1$ is combined with its neighbor $D_2$ to form $10\,D_1 + D_2$ in the first set of adders. The interim sum $S_1$ which results then appears at the input of the next set




PARALLEL DATA STREAM BCD TO BINARY CONVERTER

84-1



FIGURE 7.

of adders where it is in turn combined with the next most significant digit. In this way the combining processes ripple through the sets of adders until the completely converted number is finally gated out through the Output Buffer.

4.3.7 A circuit design for the serial converter is shown in Figure 8. Figure 9 shows the circuit symbols and conventions used in this report. To estimate the size of this converter the concept of a normalized module is employed. A normalized module is a printed circuit board that may contain any of the following functional units:

| | |
|---|---|
| 1/4 | Bit Decoder |
| 1/3 | Divider Network |
| 1/2 | Combined X4 Multiplier and Adder |
| 1 | Full Binary Adder |
| 2 | Half Binary Adders |
| 4 | Flip Flops |
| 10 | NOR Gates |
| 10 | Drivers |

For purposes of calculating circuit delay, the converter shall be built of circuitry that does not introduce more than 0.1μ sec gating delay per stage and the cycle time for the flip flops shall not exceed 0.2 μsec. These circuit requirements are easily met with materials currently available at reasonable cost.

| | 6 Decade | ENM | 9 Decade | ENM |
|---|---|---|---|---|
| Input Buffer Flip Flops | 12 | 3 | 12 | 3 |
| Control Flip Flops | 12 | 3 | 12 | 3 |
| Clock | 1 | 1 | 1 | 1 |
| NOR Gates | 10 | 1 | 10 | 1 |
| Hexidecimal Decoder | 1 | 1 | 1 | 1 |
| Addend Register Flip Flops | 4 | 1 | 4 | 1 |
| Augend Register Flip Flops | 17 | 5 | 27 | 7 |
| Sum Buffer Flip Flops | 20 | 5 | 30 | 7.5 |
| Stream Adders/Full Adders | 20 | 20 | 30 | 30 |
| Drivers | 10 | 1 | 10 | 1 |
| Equivalent Normalized Modules | | 41 | | 56 |
| Conversion Delay | 25 μsec | | 56 μsec | |

4.3.8 Figure 3 demonstrates the quantity of circuitry required vs. conversion delay in series and parallel converter operations capable of accomodating from two to nine decade numbers.

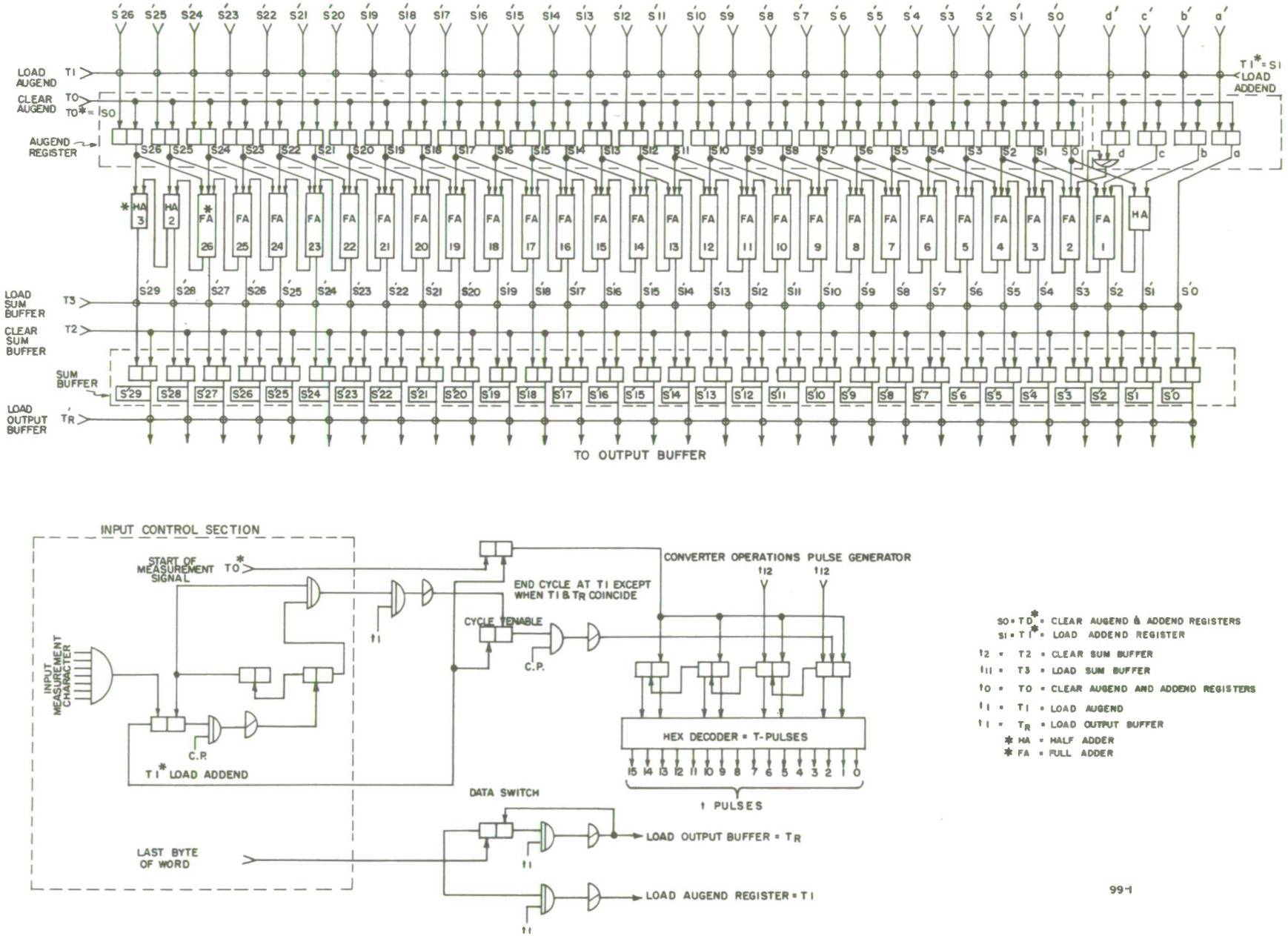



FIGURE 8. SERIAL BY BCD CHARACTER, BCD TO BINARY CONVERTER

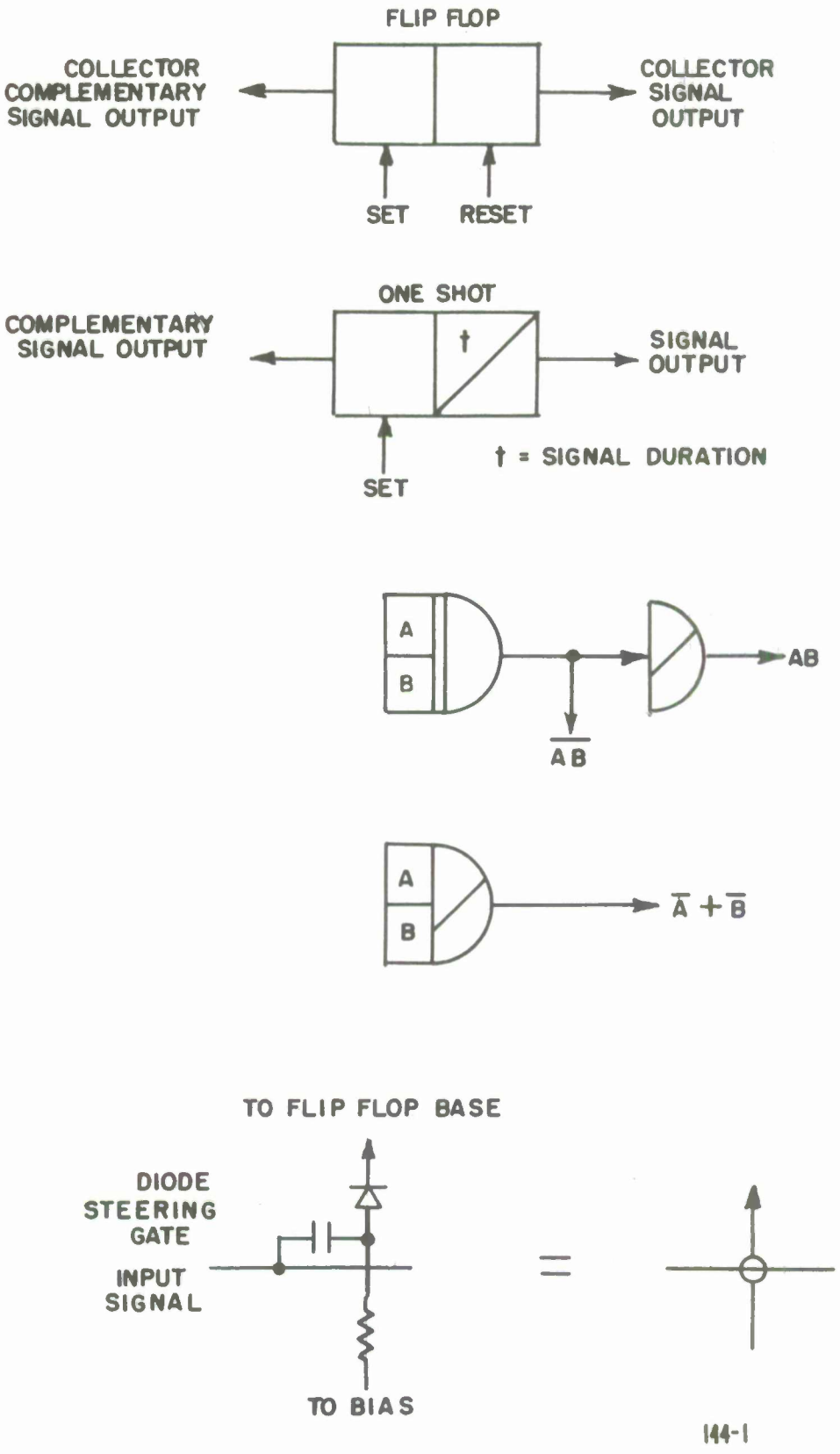



FIGURE 9. CIRCUIT SYMBOLS AND CONVENTIONS

4.3.9 The tally of circuit components required for parallel conversion is as follows:

| | 6 Decade | ENM | 9 Decade | ENM |
|---|---|---|---|---|
| Converter Control | | | | |
| Clock | 1 | 1 | 1 | 1 |
| Flip Flops | 8 | 2 | 8 | 2 |
| Hexidecimal Decoder | 1 | 1 | 1 | 1 |
| NOR Gates | 10 | 1 | 10 | 1 |
| Converter | | | | |
| Word Buffer Flip Flops | 24 | 6 | 36 | 9 |
| NOR Gates | 5 | 1 | 8 | 1 |
| Half Adders | 15 | 8 | 24 | 12 |
| Full Adders | 48 | 48 | 116 | 116 |
| Drivers | 4 | 1 | 4 | 1 |
| Equivalent Normalized Modules | | 69 | | 144 |
| Conversion Delay | 4 μsec | | 6 μsec | |

4.3.10 Referring again to Figure 3 it can be seen that the serial converter can meet the conversion speed requirement. For a data precision of one part per million or less the parallel conversion technique is competitive with the serial conversion technique in the amount of circuitry required. At a precision of one part per billion, however, the serial conversion method requires only about 40% of the amount required for parallel conversion. In terms of time required for conversion, the parallel conversion method is much more rapid.

4.4 BINARY TO BCD CONVERSION.

4.4.1 Since a 30 bit binary number converts to 9 decades in decimal or 36 bits in Binary Coded Decimal, the incoming message structure must contain 6 bits of zeros or other expendable data to permit the input and output transmission rates to be the same. There are at least three feasible methods for converting data from binary to BCD. In all of these methods the more significant bits are processed first.

4.4.1.1 Method 1. The decimal equivalent of each binary bit is summed in a decimal accumulator.

4.4.1.2 Method 2. A binary number up to and including 1001 (the equivalent of 9 in decimal numbers) may also be considered as a BCD number. This method takes advantage of this by dividing the data word into BYTES containing up to 3 bits (octal). By treating these BYTES as decimal numbers and by using decimal operations, the binary number is converted to a BCD representation. Each BYTE then would be ranked according

to powers of the radix. In the general case, the radix referred to may differ from 2 according to whether the data word is converted by bit or by BYTE. As in the above case, when the data word is subdivided into fixed length BYTES, the radix has a value of two raised to the power that corresponds to the number of bits in the BYTE. Conversion is effected by using decimal operations in the following recursive iterations: The first cycle is completed when the most significant BYTE is multiplied by the value of the radix while the next most significant BYTE is added to the product. The succeeding iterations are performed by multiplying the resulting sum obtained in the preceding iteration by the value of the radix and adding the next most significant BYTE outstanding to the result. This procedure is terminated when the least significant BYTE is finally included in the resulting sums.

4.4.1.3 Method 3. This method is a casting out of tens process. The data word and the quotients obtained are divided by ten to yield successive remainders whose values correspond to the decimal digits in increasing powers of ten.

4.4.2 A brief description of the implementation of each method follows.

4.4.2.1 Implementation of Method 1. In this method the BCD equivalent of each binary bit of a data word is summed by decimal adders. The following description refers to Figures 10 through 13.

4.4.2.1.1 The Start of Message Signal ($S_0$ in Figure 10) clears the Augend Register, the Data Character Buffer, and the Data Bit Counter at time $T_0$. The first data character is loaded by signal $S_1$ into the 6-bit shift register called the Data Character Buffer, to initiate the conversion process at time $T_1$. The major bit of the Data Character Buffer and the number stored in the Data Bit Counter determine the value of the BCD number serving as the Addend to the decimal adders. The Bit to BCD Equivalent Decoder is used to generate the proper BCD equivalent for each binary bit. This number and the BCD number stored in the Augend Register are added together by the stream decimal adders (see Figure 11). While these quantities are being added together, the Sum Buffer is cleared at the generated $T_2$. The sum is stored in the Sum Buffer at $T_3$. The Augend Register is cleared at $T_0$ to complete the conversion of the first bit to BCD. At $T_1$, a new addition cycle is initiated when the Augend Register is loaded with the contents of the Sum Buffer; the Data Character Buffer has its contents shifted one bit position to the left; and the Data Bit Counter is increased by one. When all 6 bits of the first character have been processed, the conversion can proceed only when the next character is loaded into the Data Character Buffer. This action may occur at $T_1$ for uninterrupted conversion; otherwise, the conversion halts until it does. If this is the case, the status of the converter is: The

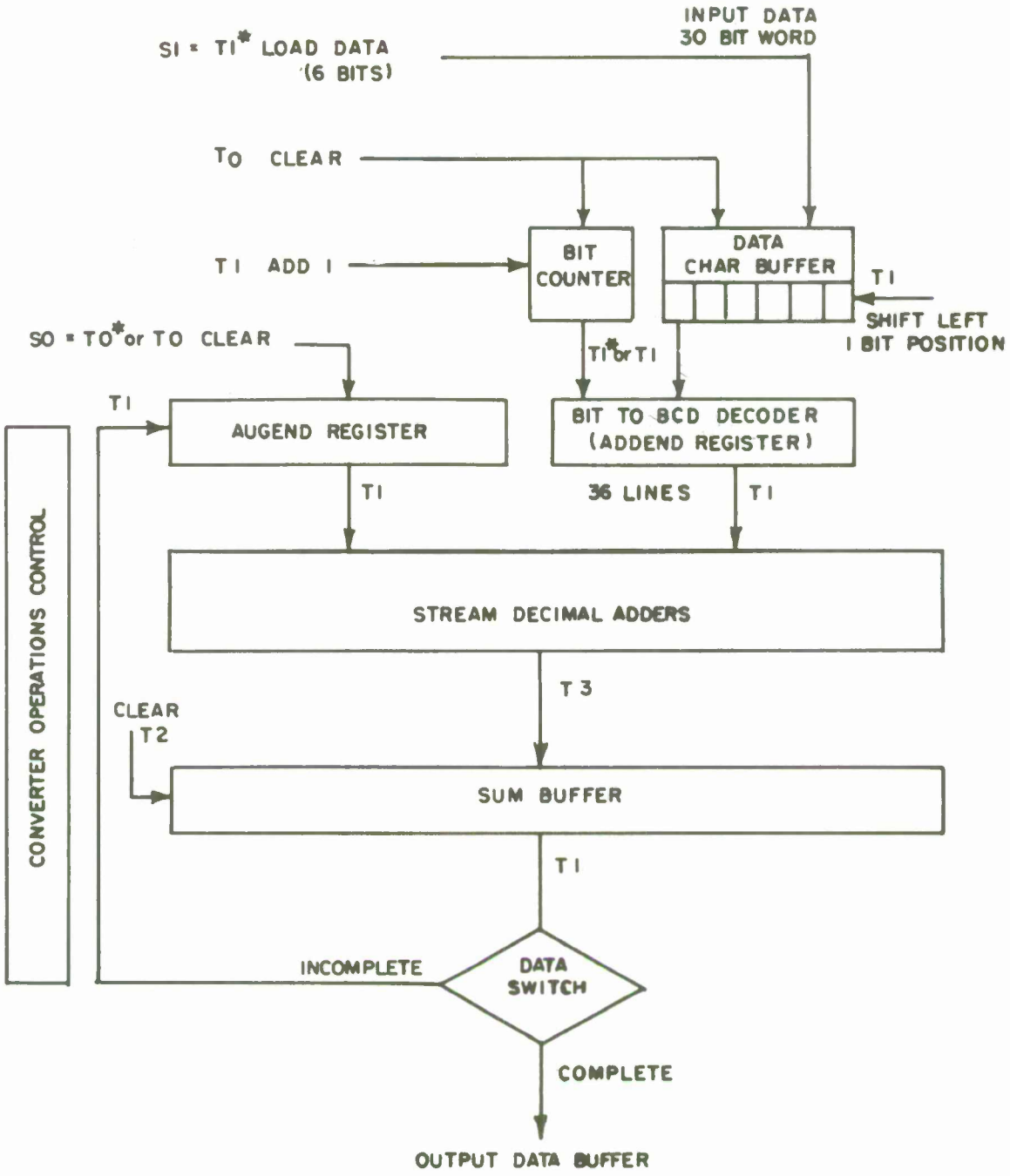



FIGURE 10. SERIAL BY BIT STREAM BINARY – BCD CONVERTER

85-1

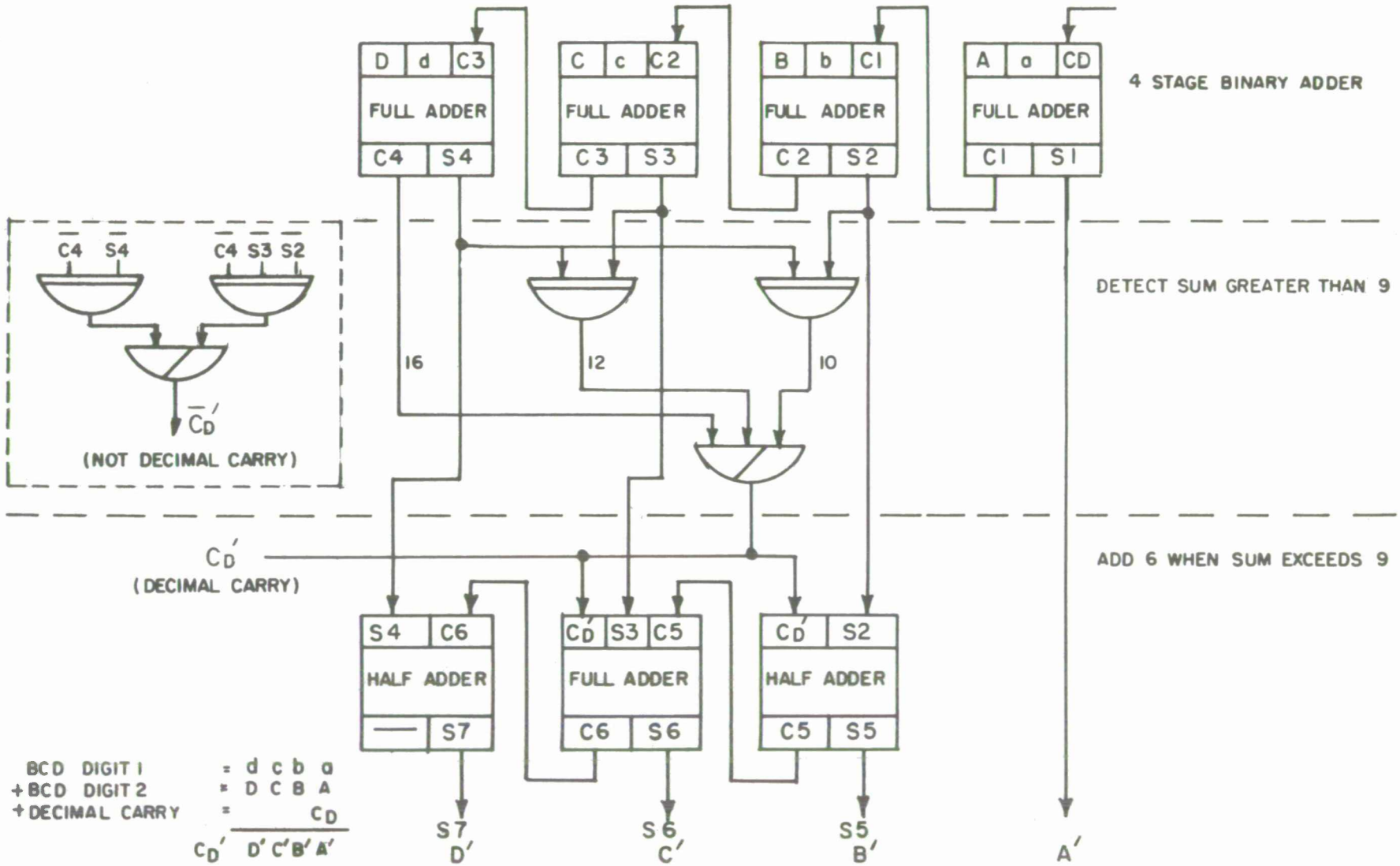



FIGURE 11. FULL DECIMAL ADDER

145-1

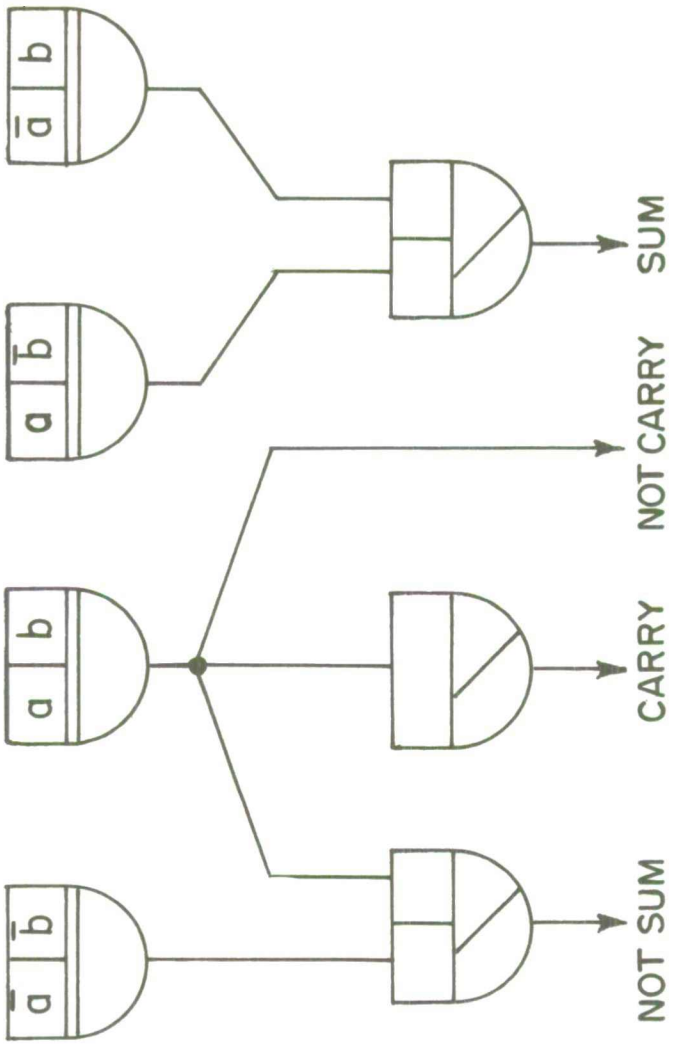
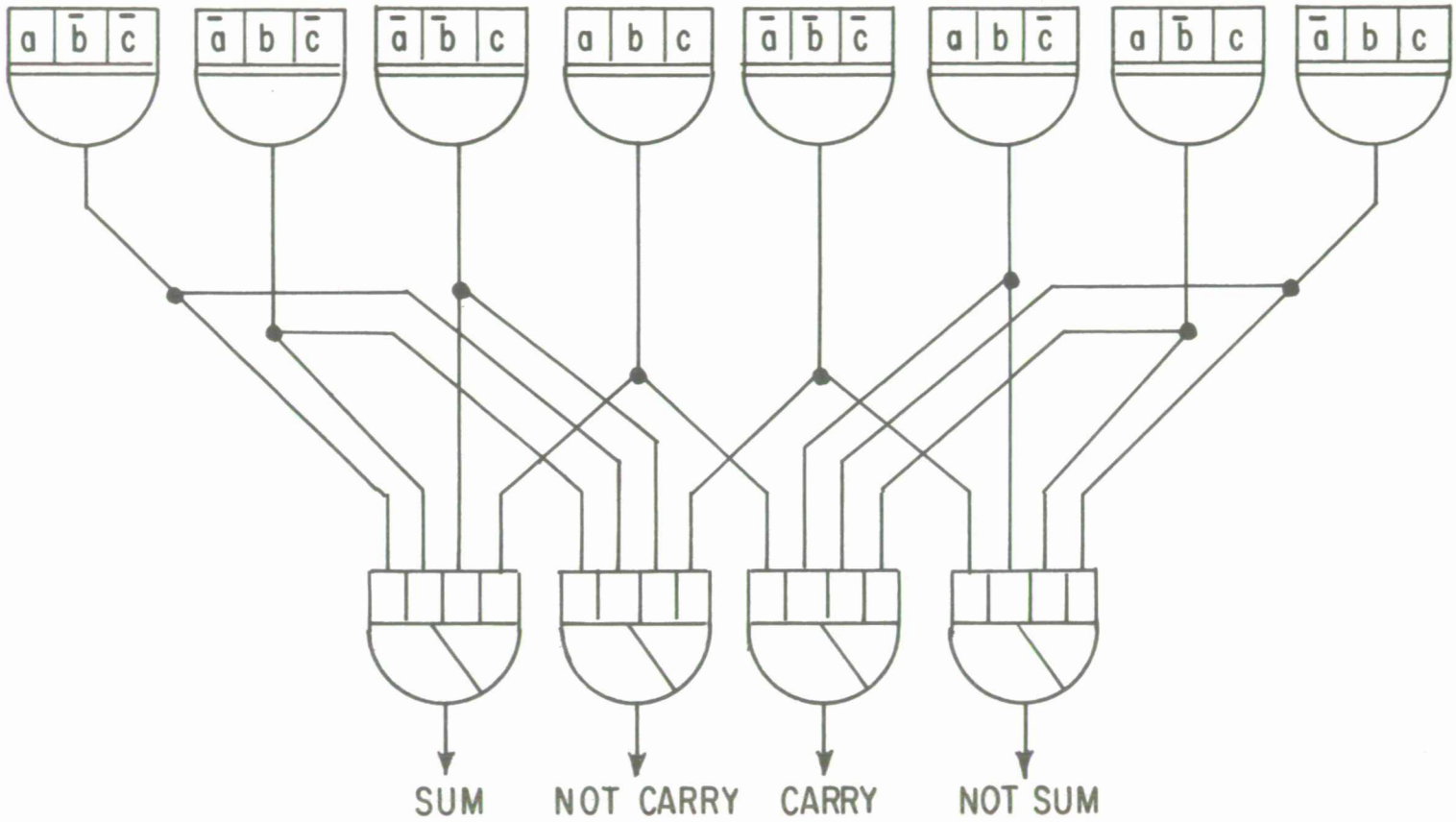



FIGURE 12. TWO LEVEL HALF-ADDER

147-1




148-1

FIGURE 13. TWO LEVEL FULL-ADDER

Augend Register is storing the BCD equivalent of the first character, the Data Character Register is empty, and the Data Bit Counter stores a count of six. The succeeding characters are processed in the same way as the first. While the fifth data character is being processed, and after the Data Bit Counter has been incremented to a count of 29, at $T_2$ the Data Switch is set to the "Conversion Completed" state. As a consequence, when the final bit has been processed and the Augend Register has been cleared at $T_0$ the data in the Sum Buffer is loaded to the Output Buffer, the Data Bit Counter is cleared, and the Data Switch reset to the "Conversion Not Complete" state. The converter may now accept the first character of the next data word for processing.

4.4.2.1.2 One measure of the practicality of this method for conversion is the calculation of the conversion speed, which is obtained by multiplying the gating delay per stage by the total number of gating levels involved for the entire conversion. The gating delays are introduced by the decimal adders employed in the method. The type chosen is a compromise between the quantity of circuitry required and the delay with which the decimal carry and BCD sums are generated. The two BCD digits and the decimal carry from the preceding decade are added together in a two stage process to generate a BCD sum and decimal carry. First these quantities are added together as binary quantities (Figure 11). Then six is added to the result when the sum exceeds 9. By this means the BCD sum and decimal carry are generated by gating through 16 and 10 levels of logic, respectively. Half adders and full adders of the type shown in Figures 12 and 13 are used because they can generate the sum, the carry, and their respective complements, by gating through only two levels of logic. Using the established circuit design parameters of 0.1$\mu$sec gating delay per stage and 0.2$\mu$sec flip flop cycle time, the conversion delay for a 30 bit binary is 294 sec. This is nearly five times the acceptable limit of 60$\mu$sec. To achieve the speed desired using this method required circuitry having less than 0.02$\mu$sec gating delay per stage. Within the techniques and materials currently available this type of response cannot be easily or economically achieved and is therefore considered to be an impractical approach. The number of addition cycles can be reduced by the pairing of certain bits so that their BCD equivalents are superposed to form a composite BCD equivalent. However, only a 20% time reduction can be achieved and this only at the cost of assembling the entire data word prior to conversion and greatly complicating the control circuitry.

4.4.2.1.3 The method described above requires the following circuitry:

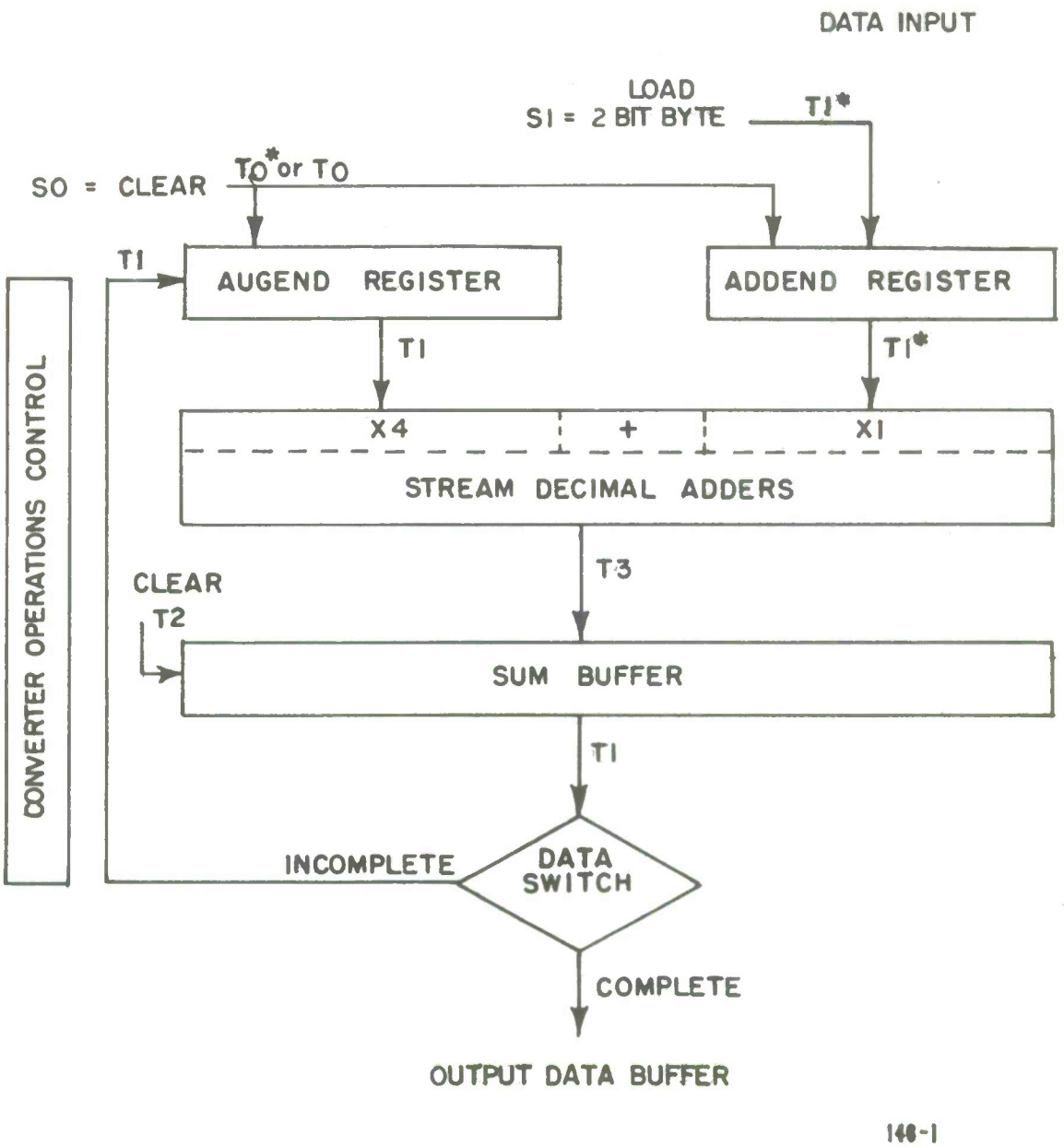


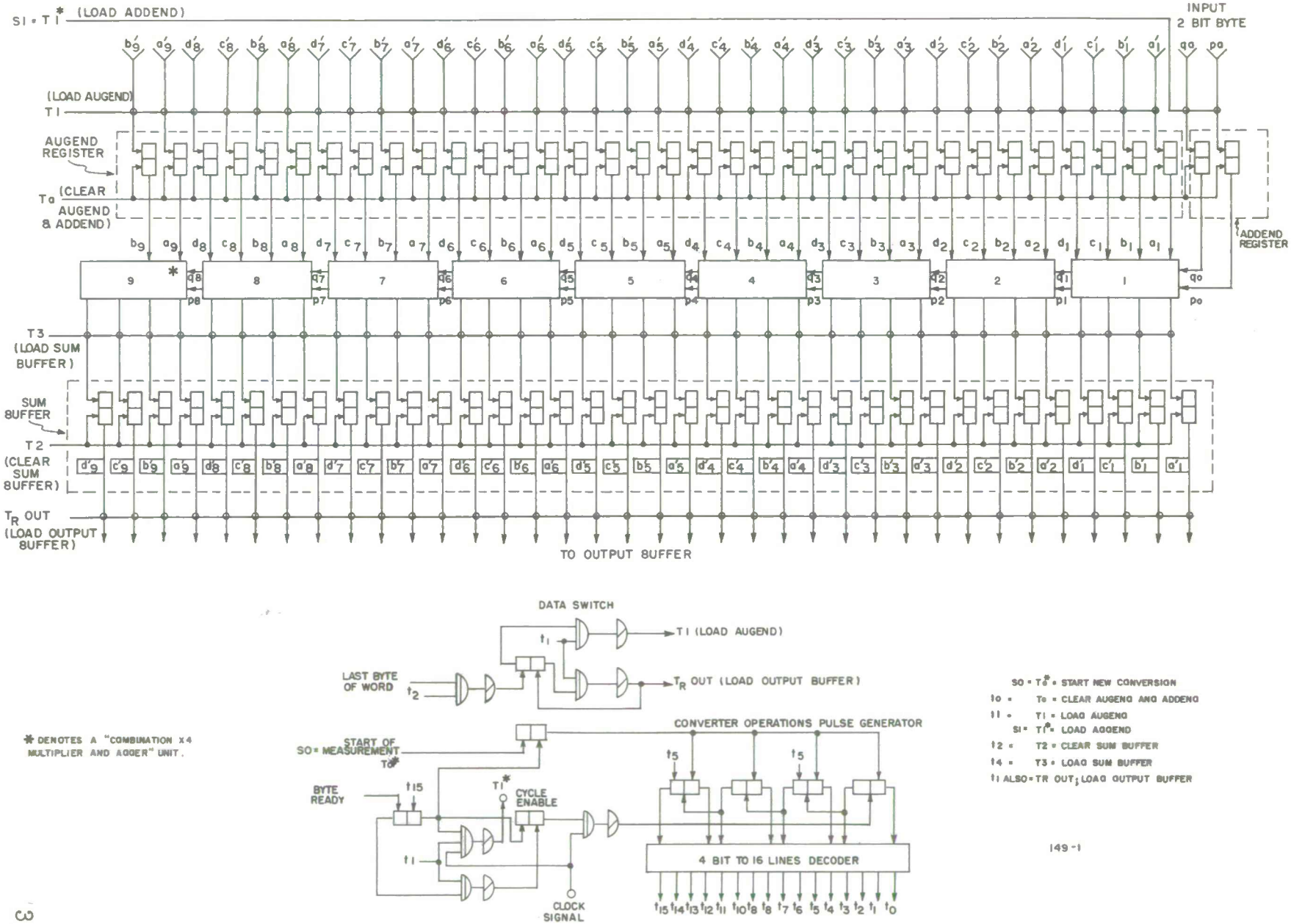

FIGURE 14. SERIAL DATA STREAM BINARY-BCD CONVERTER

| | 30 Bit Data Word | ENM | 20 Bit Data Word | ENM |
|---|---|---|---|---|
| Decimal Adders | 9 | 9 | 6 | 6 |
| Flip Flops: | 36 | 9 | 24 | 6 |
| Augend Sum Buffer | 36 | 9 | 24 | 6 |
| Data Character Buffer | 6 | 1.5 | 6 | 1.5 |
| Data Bit Counter | 5 | 1.25 | 5 | 1.25 |
| Bit to BCD Equivalent Matrix | 1 | 4 | 1 | 4 |
| Drivers | 10 | 1 | 7 | 1 |
| Control | 6 | 6 | 6 | 6 |
| Equivalent Normalized Modules | | 31 | | 22 |

4.4.2.2 Implementation of Method 2. To implement this method, it is expedient to divide the incoming data word into two bit (quantenary) characters as a compromise between the complexity of the required circuitry and the number of required iterations.

4.4.2.2.1 The Start of Message Signal $S_0$ (see Figures 6, 14, 15, and 16) starts the conversion process by clearing the Augend and Addend Registers at time $T_0$. The Addend Register is loaded with the input standardized quantenary BYTE by the signal $S_1$ at the generated $T_1$. This starts the process of multiplying the contents of Augend Register by four (the radix value of a two bit BYTE), and adding to it the contents of the Addend Register. While this action is still going on at $T_2$, the Sum Buffer is cleared and at $T_3$ the Sum Buffer is loaded with the results of the summation process. At the next self-generated $T_0$ time the Augend Register and the Addend Register are cleared, and at the next self-generated $T_1$ the Augend Register is loaded with the contents of the Sum Buffer. In coincidence with the self-generated $T_1$ signal, a Load Addend signal may occur to start the next iteration. If it does not, the converter halts to await the Load Addend Signal ($S_1$) to start the next iteration. The succeeding iterations are performed as described until the last BYTE has been converted; then after clearing the Augend and Addend Registers, the converted data being held in the Sum Buffer is transferred to the Output Buffer instead of to the Augend Register. The converter would then be in the proper condition to process properly the next data word. Figure 15 shows the circuitry required for the implementation of this method.

4.4.2.2.2 To multiply the contents of the Augend Register decimally by 4 and add to it the contents of the Addend Register, a BCD combination multiplier and adder circuitry has been designed. Shown in Figure 16a and 16b are the BCD Truth Tables and the logic required to generate this function, which is obtained by gating through only 4 levels of logic. Each decade of the Augend in conjunction with a possible two bit carry from the preceding decade or Addend character is used to gate out a decimal







* DENOTES A "COMBINATION X4 MULTIPLIER AND ADDER" UNIT.

SO = T0* = START NEW CONVERSION
t0 = T0 = CLEAR AUGEND AND ADDEND
t1 = T I = LOAD AUGEND
SI = T I* = LOAD ADDEND
t2 = T2 = CLEAR SUM BUFFER
t4 = T3 = LOAD SUM BUFFER
t1 ALSO = TR OUT; LOAD OUTPUT BUFFER

149-1

FIGURE 15. SERIAL BY QUATENARY CHARACTER. BINARY TO BCD CONVERTER

BCD TRUTH TABLE FOR FUNCTION

$$4(dcba) + qp = 10\ q'p' + d'c'b'a'$$

| INPUT DATA | | | | | | OUTPUT DATA | | INPUT DATA | | | | | | OUTPUT DATA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | c | b | a | q | p | q'p' | d'c'b'a' | d | c | b | a | q | p | q'p' | d'c'b'a' |
| 0 | 0 | 0 | 0 | 0 | p | 0 0 | 0 0 0 p | 0 | 0 | 0 | 0 | 1 | p | 0 0 | 0 0 1 p |
| 0 | 0 | 0 | 1 | 0 | p | 0 0 | 0 1 0 p | 0 | 0 | 0 | 1 | 1 | p | 0 0 | 0 1 1 p |
| 0 | 0 | 1 | 0 | 0 | p | 0 0 | 1 0 0 p | 0 | 0 | 1 | 0 | 1 | p | 0 1 | 0 0 0 p |
| 0 | 0 | 1 | 1 | 0 | p | 0 1 | 0 0 1 p | 0 | 0 | 1 | 1 | 1 | p | 0 1 | 0 1 0 p |
| 0 | 1 | 0 | 0 | 0 | p | 0 1 | 0 1 1 p | 0 | 1 | 0 | 0 | 1 | p | 0 1 | 1 0 0 p |
| 0 | 1 | 0 | 1 | 0 | p | 1 0 | 0 0 0 p | 0 | 1 | 0 | 1 | 1 | p | 1 0 | 0 0 1 p |
| 0 | 1 | 1 | 0 | 0 | p | 1 0 | 0 1 0 p | 0 | 1 | 1 | 0 | 1 | p | 1 0 | 0 1 1 p |
| 0 | 1 | 1 | 1 | 0 | p | 1 0 | 1 0 0 p | 0 | 1 | 1 | 1 | 1 | p | 1 1 | 0 0 0 p |
| 1 | 0 | 0 | 0 | 0 | p | 1 1 | 0 0 1 p | 1 | 0 | 0 | 0 | 1 | p | 1 1 | 0 1 0 p |
| 1 | 0 | 0 | 1 | 0 | p | 1 1 | 0 1 1 p | 1 | 0 | 0 | 1 | 1 | p | 1 1 | 1 0 0 p |

d c b a = INPUT BCD (binary coded decimal) digit
q p = INPUT BCD carry or 2 bit BYTE
q'p' = OUTPUT BCD carry
d'c'b'a' = OUTPUT BCD digit
d c b a and d'c'b'a' range in value between 0 and 9

ALTERNATE DESCRIPTION OF FUNCTION

```
  d c b a
  d c b a
  d c b a
    q p
-----------
q'p' d'c'b'a'
```

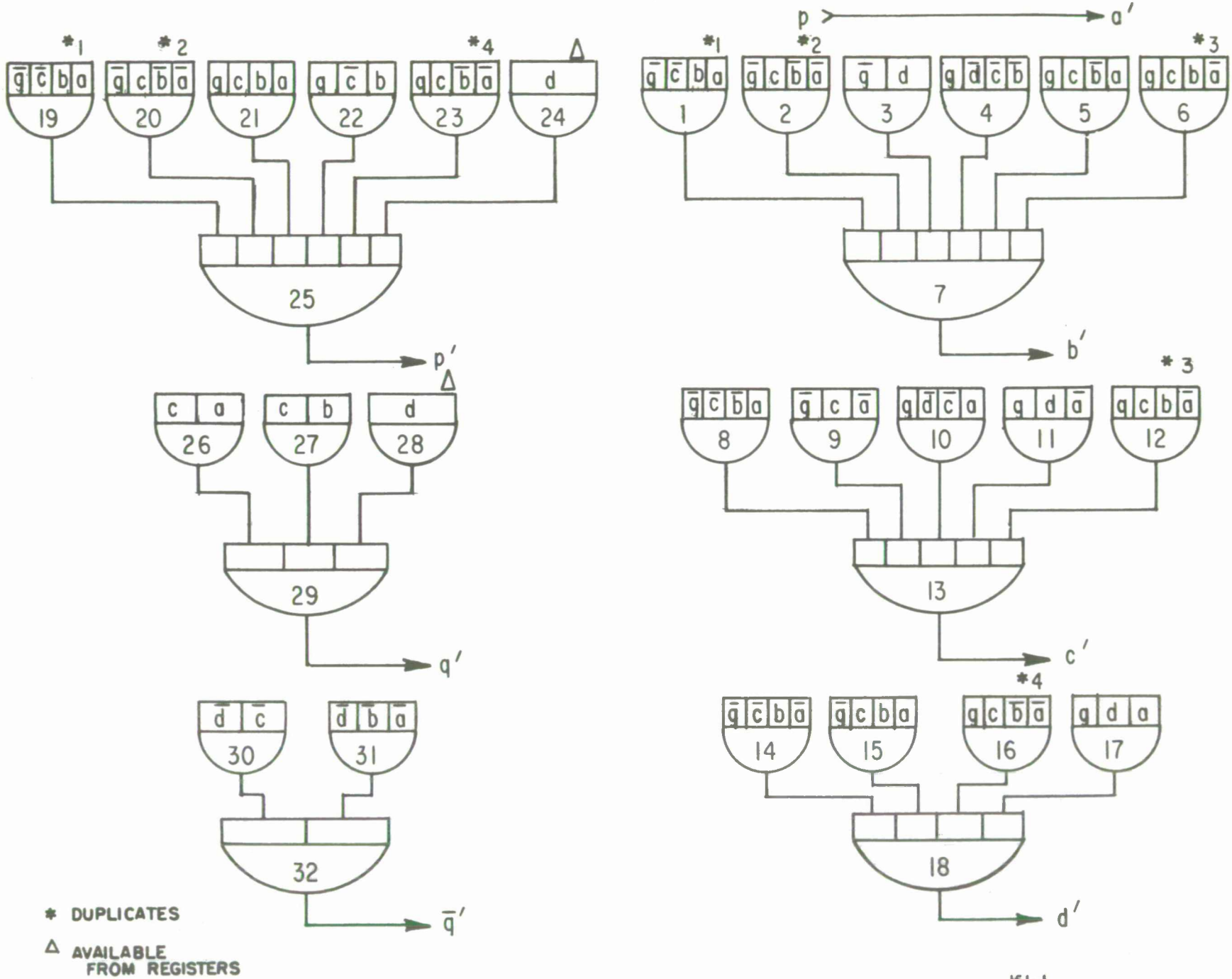
EXAMPLE:

d c b a = 0 1 1 0 = 6
q p = 1 1 = 3
then q'p' = 1 0 = 2
d'c'b'a' = 0 1 1 1 = 7

since $4 \cdot 6 + 3 = 27$

150-1

FIGURE 16a. TRUTH TABLE FOR THE BCD X4 MULTIPLIER AND ADDER




151-1

FIGURE 16b. BCD COMBINATION X4 MULTIPLIER AND ADDER

sum, and at most, a two bit carry. This is so, since a series of nines in the Augend Register when multiplied by four and having a carry of three added to it can at most generate a sum of 9 and a carry of three. This effect results in limiting the carry from any particular decade to the next higher decade. At most, it can effect the least significant bit of the decade beyond, making it odd or even. Since the propagation of carries is limited, conversion speed is high. Furthermore, the circuitry required is little more than required for 2 binary full adders.

4.4.2.2.3 The estimated time for conversion can be calculated by noting that two bits are processed at a time. To process 30 bits, 15 iterations are required. Assuming that each iteration can be completed in the time to gate through 4 levels of logic, the complete conversion requires 15 times this value (60 levels) or 6.0 μsec plus 15 storage cycles or 3 μsec for a total conversion delay of 9 μsec. This more than meets the design goal for conversion speed.

4.4.2.2.4 Study favors this method since the circuit requirements are reasonable and their quantity and complexity are considered optimal. This technique also results in the shortest conversion time for circuitry operating at any given speed. This method requires the following circuitry:

| | 30-Bit Word | ENM | 20-Bit Word | ENM |
|---|---|---|---|---|
| Control Circuitry | 7 | 7 | 7 | 7 |
| Combined X4 Multiplier and Adders | 9 | 18 | 6 | 12 |
| Flip Flops: Augend | 36 | 9 | 24 | 6 |
| Addend | 2 | 0.5 | 2 | 0.5 |
| Sum Buffer | 36 | 9 | 24 | 6 |
| Drivers | 10 | 1 | 10 | 1 |
| Equivalent Normalized Modules | | 45 | | 33 |

In Figure 4 is shown the module count and the conversion delays for Method 2 as a function of the reciprocal data precision.

4.4.2.3 Implementation of Method 3. The entire data word is transferred to the Dividend Register before the conversion process is initiated (see Figure 17). When the data word has been transferred it is divided by ten by the divide networks and the remainder obtained stored in the Decimal Digit Register. The quotient obtained replaces the original data word to initiate the next and similar division cycle. When 9 division cycles have been performed, the Decimal Digit Register contains the 9 BCD characters equivalent of the original 30 bits. To facilitate the conversion process and optimize the conversion speed by a reasonable quantity of circuitry, the data word is processed piecemeal. For this purpose the data word

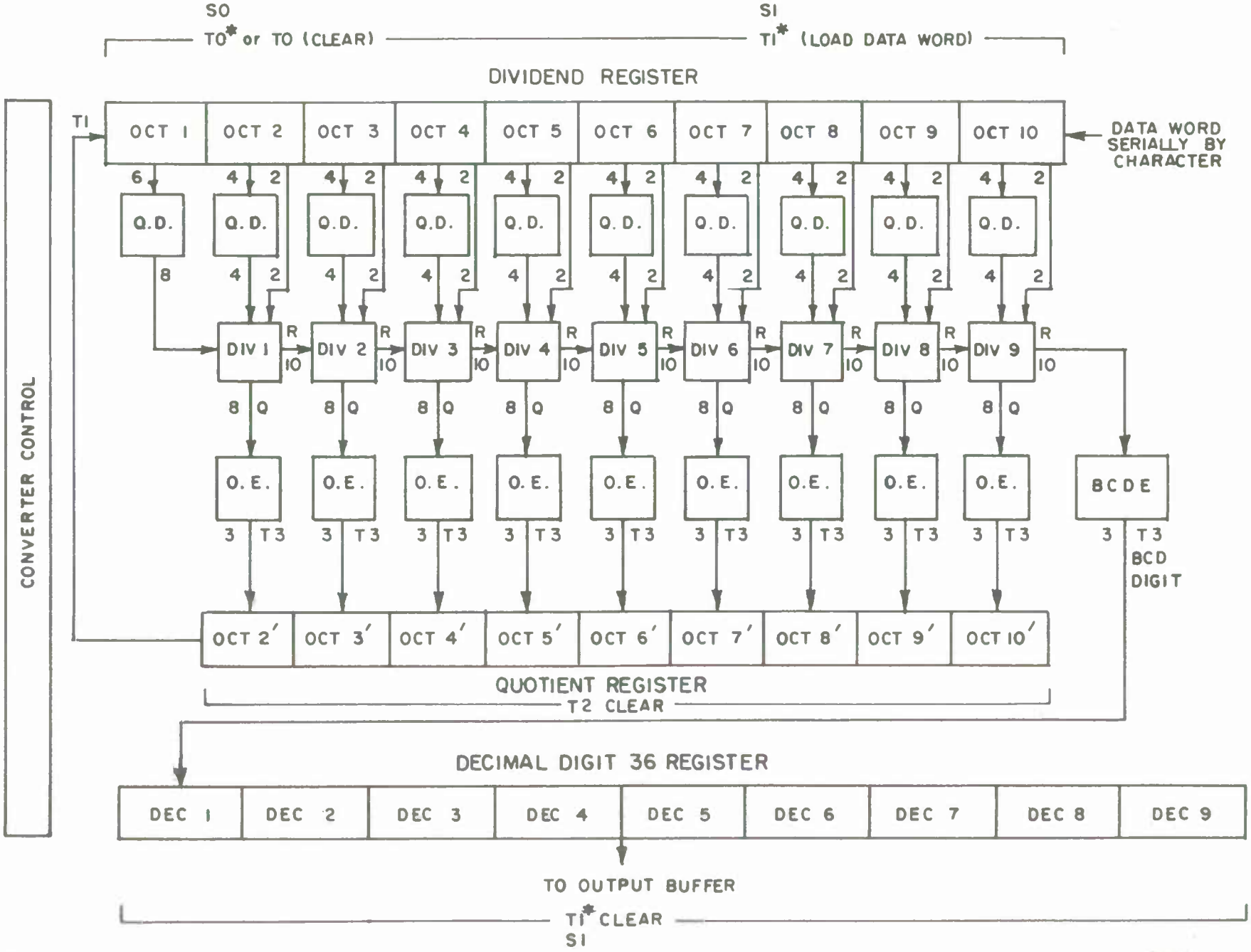



FIGURE 17. BINARY TO BCD CONVERSION BY RECURSIVE DIVISION

152-1

is subdivided into octal (3 bit) characters and the divider network constrained to operate upon 7 bits; four bits being the remainder from the preceding divider network output and three bits from the Dividend Register. The remainder from the preceding divider network, serving as one input to the following divider network is decoded to activate one of ten lines. The major two bits of the octal character serving as the other input to the divider network are decoded to activate one out of 4 lines by the Quantenary Decoder (QD). The first divider network accepts as one input the most significant octal character decoded into one out of 8 lines, and the decoded next most significant octal character as its other input. Since the least significant bit of the octal character merely makes the remainder generated odd or even and does not affect the quotient, it is not included in the division process. Thus, the divider network has 10 input lines corresponding to the value of the remainder from the preceding divider network and 4 lines corresponding to the major two bits of the octal character from the Dividend Register. There are 40 possible ways of activating these lines and the appropriate value of the quotient is gated out on one of 8 lines and appropriate value of the remainder on one of 5 lines. The 5 remainder lines are combined logically with the least significant bit of the input octal character to activate one out of ten lines to serve as one input to the next divider network.

4.4.2.3.1 The Start of Message Signal $S_0$ clears the Dividend Register. The $S_1$ signal loads the Dividend Register with a data word, clears the Decimal Digits Register with the generated $T_1$, and initiates the conversion process. The data word is subdivided into octal characters which, after being decoded by their respective decoders, serve as inputs to the divider network chain. A remainder, gated out by the preceding higher ranking divider network, serves as the other input. While these divider networks are generating their quotient and remainders, the Quotient Register is cleared by the generated $T_2$. The quotient from each divider network is encoded into an octal character by the Octal Encoders (OE), and the remainder from the lowest ranking divider network is encoded into a BCD character by the BCD encoder (BCD E). When $T_3$ is generated the Quotient Register is loaded with the quotient and the Decimal Digits Register is loaded with the BCD character generated by the last divider network. At cycle time $T_0$ the Dividend Register is cleared and at $T_1$ it is loaded with the contents of the Quotient Register to initiate the next division cycle. The succeeding division cycles are performed in the same way as the first; and at the end of the ninth cycle after the Decimal Digits Register is loaded with the last BCD character, the content of that register is loaded into the Output Buffer. The next $S_1$ signal loads the next data word into the Dividend Register, clears the Decimal Digits Register, and initiates a new conversion process. Each iteration is completed in about the time required to gate through 20 levels of logic, and the entire conversion in 9 such iterations for a total propagation delay due to 180 levels

of logic, or 18 μsec plus 9 storage cycles of 1.8 μsec for a total conversion delay of 19.8 μsec. This method therefore meets the design goal for conversion speed.

| | 9 Decade | ENM | 6 Decade | ENM |
|---|---|---|---|---|
| Divider Matrices | 9 | 27 | 6 | 18 |
| Octal Decoders | 1 | 1 | 0 | 0 |
| Quantenary Decoders | 9 | 4.5 | 7 | 3.5 |
| Octal Encoders | 9 | 4.5 | 6 | 3 |
| BCD Encoders | 1 | 1 | 1 | 1 |
| Dividend Register Flip Flop | 30 | 7.5 | 20 | 5 |
| Quotient Register Flip Flop | 27 | 7 | 18 | 4.5 |
| Decimal Digits Register Flip Flop | 36 | 9 | 20 | 5 |
| Drivers | 10 | 1 | 7 | 1 |
| Equivalent Normalized Modules | | 52 | | 41 |

4.4.2.3.2 While this method is a satisfactory one for converting binary numbers to decimal, it is, however, more complex, requires more circuitry, and is only half as fast as conversion Method 2. It is, therefore, not as desirable.

4.4.3 Parallel Conversion of Binary to BCD. In parallel (or broadside) conversion the entire data word is converted in one continuous stream conversion process. (See Figure 18). The first two 2 bit BYTES are combined by multiplying the most significant 2 bit BYTE by four and adding in the other 2 bit BYTE using the Combined X4 Multiplier and Adders as previously described in Method 2. The result of this operation is presented as the input to the next level of combination multiplier and adders where the process is repeated using the next most significant 2 bit BYTE. This process continues using additional groups of combination multipliers and adders until the entire data word is processed, the final result being the required BCD equivalent of the 30 bit binary number. The required amount of circuitry is:

| | 20 bit number | ENM | 30 bit number | ENM |
|---|---|---|---|---|
| Word Buffer Flip Flops | 20 | 5 | 30 | 7.5 |
| Combination X4 Multiplier and Adders | 32 | 64 | 71 | 142 |
| Drivers | 2 | 0.2 | 3 | 0.3 |
| Equivalent Normalized Modules | | 70 | | 150 |
| Conversion Delay | 2.5 μsec | | 4.4 μsec | |

In Figure 4 is shown the module count and conversion delay for parallel conversion as a function of reciprocal data precision. In comparing the

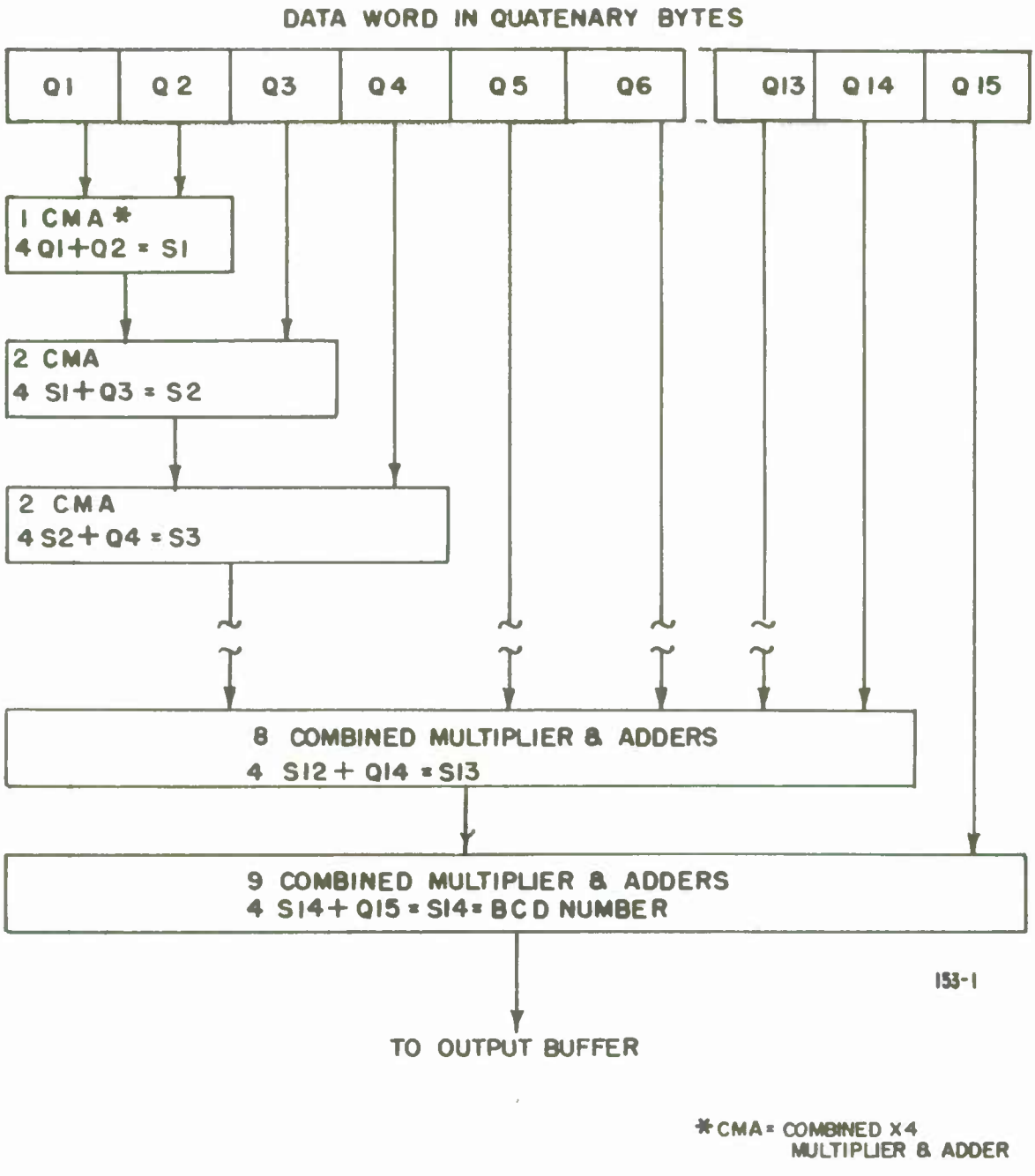



FIGURE18.PARALLEL DATA STREAM BINARY TO BCD CONVERTER

factors involved in this technique with those associated with the serial methods previously discussed and again referring to Figure 4, it is evident that the latter is considerably more economical in circuit requirements at the expense of only a small increase in conversion delay. However, processing parallel data by serial methods is no problem since it can be stored in the Input Buffer from which it can be presented to the Converter in virtually any form desired. Since this can readily be accomplished, it is considered that it is unnecessary to further consider binary to BCD conversion of parallel inputs.

4.4.4 In summary, of the three methods described, Method 2 is the most attractive in regard to conversion speed, circuit simplicity, and the quantity of circuitry required. Parallel word conversion is to be accomplished by "shifting" out of the Input Buffer since direct parallel conversion is impractical.

## 4.5 CONVERSION OF DATA FROM TIME IN HOURS, MINUTES, SECONDS AND MILLISECONDS TO BINARY MILLISECONDS.

### 4.5.1 General Considerations Effecting Time Conversion.

4.5.1.1 Incoming time measurement data can be expected to be in the form of BCD hours, minutes, seconds, and fractional seconds to thousandths of seconds or milliseconds. To avoid the necessity of operating with whole numbers and their associated fractions, the adopted value of the least significant binary bit shall be the millisecond. Similarly, when converting from binary milliseconds, the converted units will be in terms of hours, minutes, and milliseconds.

4.5.1.2 Each digit of a number representing time in hours, minutes, and milliseconds has associated with it either some power of 10 or 6. Consequently, the conversion process will closely parallel the optimum BCD to binary and binary to BCD conversion already discussed. It is only necessary, therefore, that the difference of techniques or circuit components be described.

4.5.1.3 As stipulated under the design criteria, the capacity of the converters under consideration was limited to a 9 decade decimal number or the equivalent of a 30 bit number. The capacity of the converter is therefore set at a maximum of 298 hours, 15 minutes, and 41.823 seconds, since this converts to one less than 2 to the 31 power which is the maximum value of a 30 bit number.

### 4.5.2 The BCD to Binary Time Converter.

4.5.2.1 Differences which exist between time conversion techniques and decimal to binary conversion techniques result from the fact that hours, minutes, and seconds of the time conversion problem are associated with a factor of 60 rather than 100. Thus, for time conversion it is first necessary to convert the BCD hours to binary hours, then to electrically adjust the stream adders in order to multiply the binary hours by 6 to obtain tens of minutes. The adder circuits are then again reconstituted to multiply binary tens of minutes by 10 to obtain binary minutes. Once again the stream adder circuits are modified to multiply binary minutes by 6 to obtain binary tens of seconds and reconverted to continue the conversion of the remaining time digits. The process can be summarized algebraically for the time data Hh hours, Mm minutes, Ss seconds as follows:

$$10\,(6\,(10\,(6\,(10H + h) + M) + m) + S) + s \ \ldots\ldots\text{etc.}$$

4.5.2.2 The multiplication by 6 as outlined above is accomplished by adding 4 and 2 times the number instead of 8 and 2 as in the straight decimal conversion. Actual modification of the stream adders for this purpose is simply achieved through the use of switch logic which couples the respective bits from the 8 to 4 position as required. Referring to Figures 5 and 6, the switch logic for controlling the multiplication factor is interposed between the Augend and Addend Registers on one hand and the stream adders on the other. Figure 19 shows the required multiplier switch logic. The required conversion time is comparable to that of the 9 decade BCD to Binary Converter. The required circuitry is:

| | Units | ENM |
|---|---|---|
| Input Buffer Flip Flops | 12 | 3 |
| Control | | |
| Flip Flops | 12 | 3 |
| Clock | 1 | 1 |
| NOR Gates | 10 | 1 |
| Hexidecimal Decoder | 1 | 1 |
| Converter | | |
| Addend Register Flip Flops | 4 | 1 |
| Augend Register Flip Flops | 27 | 7 |
| Sum Buffer Flip Flops | 30 | 7.5 |
| Stream Adders/Full Adders | 30 | 30 |
| Drivers | 15 | 2 |
| Switch Logic NOR Gates | 50 | 5 |
| Equivalent Normalized Modules | | 62 |
| Conversion Delay | | 56 μsec |

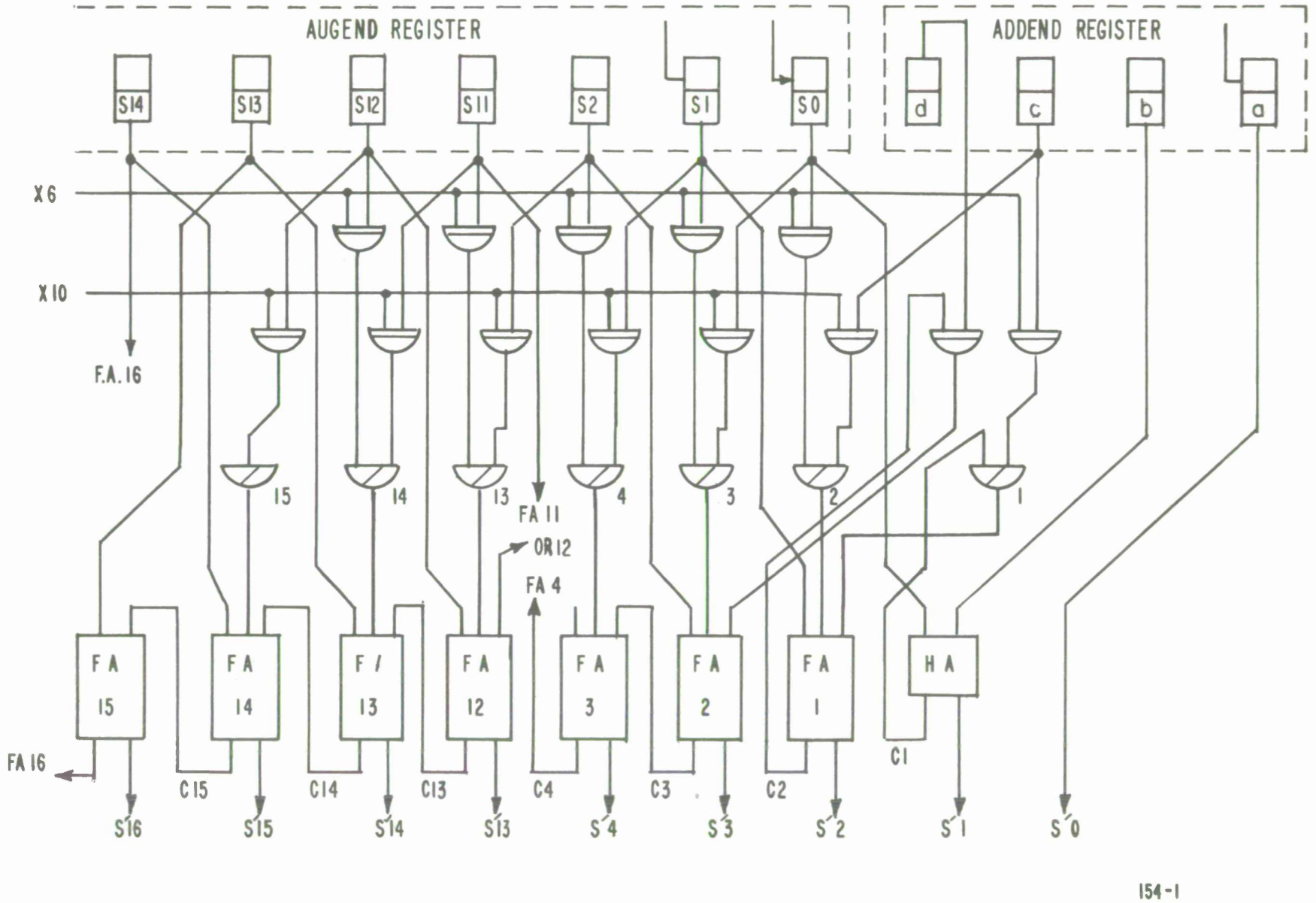



FIGURE 19. MULTIPLIER SWITCH LOGIC FOR CONVERSION OF BCD TIME TO BINARY

4.5.3 Parallel conversion is achieved through the same process as in the parallel decimal BCD to Binary Conversion (see Figure 7) except that the third and fifth groups of stream adders are constructed to multiply by 6 instead of ten. However, as in the former case, because of the increased circuitry involved in this type of conversion it is considered more practical to have the Input Buffer shift the incoming parallel data in serial form for presentation to the Radix Converter. The required amount of circuitry and conversion speed is exactly the same as for the 9 decade parallel BCD to binary converter which is listed on page 22.

## 4.6 CONVERSION OF BINARY MILLISECONDS TO BCD HOURS, MINUTES, AND MILLISECONDS.

4.6.1 The optimum procedure for converting binary milliseconds to BCD base 60 time units is to use a variation of the Method 2 described above for converting binary numbers to decimal. As before, the capacity of the converter is 30 bits of binary milliseconds or the equivalent of 298 hours, 15 minutes, and 41,823 milliseconds.

4.6.2 Whereas in the binary to decimal converter all the BCD combined X4 multipliers and adders were to the base ten, in this case the 5th and 7th stage combined multipliers and adders are to the base 6. The number in units of milliseconds is processed two bits at a time in order of decreasing significance. As the decimal equivalent approaches the 60,000 decimal milliseconds mark the substitution of the base 6 combined multplier and adders automatically adjusts the totals so that what would have been 60,000 is converted to 100,000. Thus, by constraining the capacity of the 5th decade to 5 instead of 9, the first decades are limited to units of 60,000 milliseconds or one minute. Similarly, the 7th decade is limited to 5 instead of 9, producing hours computed as multiples of 60 minutes.

4.6.3 Figures 20a and 20b show the BCD Truth Tables and the required circuitry for the base 6 combined BCD X4 multipliers and adders. The total required circuitry and speed capability is the same as that in the serial conversion Method 2 for binary decimal to BCD.

## 4.7 SUGGESTED PROCEDURES FOR THE CONVERSION OF MIXED NUMBERS.

4.7.1 When mixed numbers must be converted from one radix representation to another, the whole number portion and fractional portion usually require separate consideration. In the conversion process of mixed decimal numbers to binary, the whole number portion is converted by a process involving binary operations and multiplications by ten. The fractional portion, however, is most conveniently converted by a process employing decimal operations and multipliers of 2 or 2 raised to a small power.

BASE 6 BCD TRUTH TABLE FOR FUNCTION

$$4(dcba) + qp = 6\,q'p' + d'c'b'a'$$

| INPUT DATA | | | | | | OUTPUT DATA | | INPUT DATA | | | | | | OUTPUT DATA | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| d | c | b | a | q | p | q'p' | d'c'b'a' | d | c | b | a | q | p | q'p' | d'c'b'a' |
| 0 | 0 | 0 | 0 | 0 | p | 0 0 | 0 0 0 p | 0 | 0 | 0 | 0 | 1 | p | 0 0 | 0 0 1 p |
| 0 | 0 | 0 | 1 | 0 | p | 0 0 | 0 1 0 p | 0 | 0 | 0 | 1 | 1 | p | 0 1 | 0 0 0 p |
| 0 | 0 | 1 | 0 | 0 | p | 0 1 | 0 0 1 p | 0 | 0 | 1 | 0 | 1 | p | 0 1 | 0 1 0 p |
| 0 | 0 | 1 | 1 | 0 | p | 1 0 | 0 0 0 p | 0 | 0 | 1 | 1 | 1 | p | 1 0 | 0 0 1 p |
| 0 | 1 | 0 | 0 | 0 | p | 1 0 | 0 1 0 p | 0 | 1 | 0 | 0 | 1 | p | 1 1 | 0 0 0 p |
| 0 | 1 | 0 | 1 | 0 | p | 1 1 | 0 0 1 p | 0 | 1 | 0 | 1 | 1 | p | 1 1 | 0 1 0 p |

d c b a = INPUT BCD (binary coded decimal) digit
q p = INPUT BCD carry or 2 bit BYTE
q'p' = OUTPUT BCD carry
d'c'b'a' = OUTPUT BCD digit
d c b a and d'c'b'a' range in value between 0 and 5

ALTERNATE DESCRIPTION OF FUNCTION

$$\begin{array}{r} d\ c\ b\ a \\ d\ c\ b\ a \\ d\ c\ b\ a \\ q\ p \\ \hline q'p'\ d'c'b'a' \end{array}$$

EXAMPLE:

$$\begin{array}{rl} & d\ c\ b\ a = 0\ 1\ 1\ 1 = 3 \\ & q\ p = 1\ 0 = 2 \\ \text{then} & q'p' = 1\ 0 = 2 \\ & d'c'b'a' = 0\ 0\ 1\ 0 = 2 \end{array}$$

since $4 \cdot 3 + 2 = 6 \cdot 2 + 2$

155-1

FIGURE 20a. TRUTH TABLE FOR THE BASE 6 COMBINATION BCD X4 MULTIPLIER AND ADDER

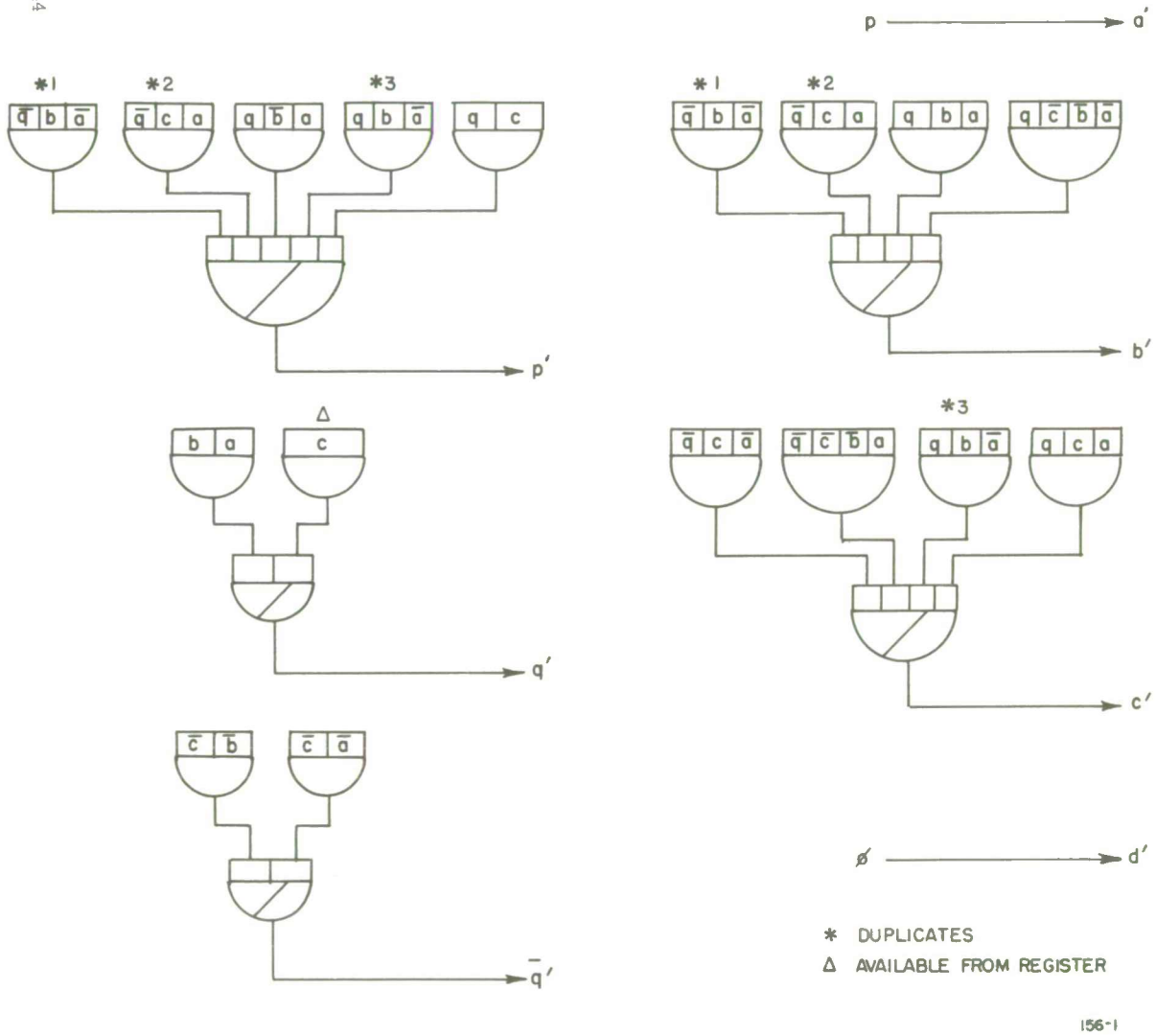



FIGURE 20b. BASE 6 BCD TIMES 4 MULTIPLIER AND ADDER

4.7.2 In the conversion process of a mixed binary number the whole number portion is most conveniently converted by a process involving decimal operations and a multiplier of 2 or 2 raised to a small power while the conversion process for the fractional portion involves binary operations and multiplications by ten.

4.7.3 The conversion process for a fraction from one radix representation to another resembles the inverse of the conversion process required for a whole number. Thus, the conversion process for decimal fractions to binary resembles the conversion process for whole numbers in binary to decimal. This resemblance also holds true for the inverse conversion operations. As a consequence it will be necessary that the translator contain essentially both the required radix converter for the whole number and inverse converter for the fractional number. There may be advantages to this since the translator by containing both types of converters can then convert mixed numbers either to binary or to decimal with relative ease.

4.7.4 Another method which may be employed to handle mixed numbers is to pre-multiply the number to be converted so that it becomes a whole number. The result will be a known power of ten or two larger than required and a simple shift in the powers of ten or two will yield the desired result. For example, in the conversion of 1.625 from decimal to binary to an accuracy of ten binary places or one part in 1024, the proper scale factor is 1024, the desired binary accuracy, divided by 1000, the decimal magnification factor. Thus 1.625 is multiplied by 1.024 which yields 1.664. When 1664 (1.664 X 1000) is converted to binary, the result is:

11010000000

which is too large by a factor of 1024. Now, moving the binary point ten places to the left (the inverse of the magnification factor) gives the correct result or 1.101. Thus it is possible to convert decimal mixed numbers to binary by using decimal operations to prescale the original number by multiplying it by a factor which is equal to the desired binary accuracy divided by the decimal magnification factor. The result when converted to binary will be too large by a factor equal to the desired binary accuracy.

4.7.5 In the conversion of binary mixed numbers to decimal, a scaling factor equal to the desired decimal accuracy divided by the binary magnification factor is chosen as above. The converted binary number will be too large by a factor equal to the desired decimal accuracy.

4.7.6 It is also possible to convert the mixed number disregarding the decimal or binary point. The converted binary result may be corrected

by multiplying it by the reciprocal of the magnification factor. Similarly, binary mixed numbers may be converted to decimal and the converted decimal number may be corrected by multiplying it by the decimal equivalent of the binary magnification factor.

4.7.7 The required pre- or post-scaling operations can be performed by the inclusion of appropriate units in the translator. In general, however, this approach is to be avoided since the scale factors become numerically unsuitable for effecting multiplication by one or two addition and shift operations. In this case true multiplication units may be necessary and these in general are expensive in terms of circuitry and processing delay. It is preferable, therefore, to follow the procedure wherein the whole number and fraction are processed separately even though this requires two types of converters.

## Section 5. CONCLUSIONS

5.1 The following conclusions are based upon the current state of the electronic art in the utilization of solid state components and the employment of flip flops as storage elements. As the requirement for storage elements increases, the circuitry requirements become extensive to the point that despite their relative high cost, read-write circuitry and core drivers become at first competitive and finally more economical than flip flops. When the full range of requirements for an in-line translator are specified in detail, it is probable that several hundred bits of storage may be required. In addition, provisions will more than likely have to be made for the storage of procedural data. Thus, the currently recommended utilization of flip flops will very probably give way to magnetic core devices for at least part of the storage components.

5.2 It appears also, that as molecular and integrated circuitry become commercially prevalent and more attractive in economy, some of the conclusions reached may need revision. In particular, as the availability of magnetic core devices increases with the attendant increases in circuitry simplicity and conversion speed, parallel conversion techniques may prove to be more attractive than serial conversion.

5.3 The following conclusions have been reached on the basis of this study and current considerations.

(1) There is a definite application for in-line translators in the range data handling network.

(2) In-line translators have the capability of achieving adequate real-time operation and can convert as fast or faster than most programmed high speed converters.

(3) In-line translators can offer a significant economic advantage for heavy flow data conversion over conversion by means of stored programs.

(4) The design and construction of in-line translators is technically feasible and the elements required for construction are commercially available at reasonable cost.

(5) In-line translators can be logically organized into basic functions which can be performed by standardized modular components.

(6) By the selection of appropriate standardized sub-elements in-line translators can be adapted to a wide range of input and output requirements.

(7) In-line translators can function without external assistance.

(8) The actual design of a complete in-line translator without stipulated input/output values for a specific application would be purely an academic exercise.

(9) The basic functionally independent sub-element of any translator will be the Radix Converter.

(10) By preconditioning input data, Radix Converters can be made system independent and therefore have universal application.

(11) Whereas only four representative types of radix converters were designed, it demonstrated the feasibility of designing virtually any type needed.

UNCLASSIFIED

Security Classification

## DOCUMENT CONTROL DATA - R&D

*(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)*

1. ORIGINATING ACTIVITY *(Corporate author)*

National Bureau of Standards
Washington 25, D.C.

2a. REPORT SECURITY CLASSIFICATION

2b. GROUP

3. REPORT TITLE

Range Data Handling Integration Studies, In-Line Translators and Binary/Decimal Converters

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(Last name, first name, initial)*

Urban Walter D.

| 6. REPORT DATE | 7a. TOTAL NO. OF PAGES | 7b. NO. OF REFS |
|---|---|---|
| November 1964 | 53 | 0 |

| 8a. CONTRACT OR GRANT NO. | 9a. ORIGINATOR'S REPORT NUMBER(S) |
|---|---|
| b. PROJECT NO. 5932 MIPR 4230486 | NBS 4230486 |
| c. | 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)* |
| d. | ESD TDR 65-195 |

10. AVAILABILITY/LIMITATION NOTICES

Qualified requesters may obtain from DDC.
Aval from OTS.

| 11. SUPPLEMENTARY NOTES | 12. SPONSORING MILITARY ACTIVITY |
|---|---|
| None | Directorate of Aerospace Instrumentation, ESD, LG Hanscom Fld., Bedford, Mass 01731 |

13. ABSTRACT

The application of In-Line Translators is covered in terms of the specific adaptation of a wired program device which can efficiently handle the translation between different numerical radices or number base codes. Design criteria, feasibility, and a functional description are presented. Since the Radix Converter is the central and promary modular element of the translator concept, the logic designs for four principal types of decimal/binary converters are described in detail.

DD FORM 1 JAN 64 1473

Security Classification

| 14. KEY WORDS | LINK A | | LINK B | | LINK C | |
|---|---|---|---|---|---|---|
| | ROLE | WT | ROLE | WT | ROLE | WT |
| Information Handling<br>Data Transmission<br>EBP<br>Data Translation<br>Decimal Converters<br>Binary Converters | | | | | | |

## INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2*a*. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2*b*. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7*a*. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7*b*. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8*a*. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8*b*, 8*c*, & 8*d*. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9*a*. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9*b*. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional